НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра автоматизованих систем обробки інформації і управління

«На правах рукопису»

УДК 004.8+025.4

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Магістерська дисертація

зі спеціальності *121 «Інженерія програмного забезпечення»*

на тему: «Математичне та програмне забезпечення класифікації наукових текстів»

Виконав: студент VI курсу, групи *ІП-82мп*

Канівець Дмитро Володимирович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *доц., доц., к.т.н., Фіногенов О. Д,* _____
(посада, науковий ступінь, , вчене звання, прізвище та ініціали) (підпис)

Консультант *доц., к.т.н., Ліщук К. І.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц., доц., к.т.н., Яблонський П. М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти *другий (магістерський) за освітньо-професійною програмою*

Спеціальність *121 «Інженерія програмного забезпечення»*
(код і назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
«___» _____201 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Канівцю Дмитру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема дисертації *Математичне та програмне забезпечення класифікації наукових текстів*

науковий керівник дисертації *к.т.н., доц. Фіногенов О.Д.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від " *28* " *жовтня* 20 *19* р. № № 3770-с

2. Строк подання студентом дисертації " *16* " *грудня* 20 *19* р.

3. Об'єкт дослідження *Бібліотечна класифікація наукових текстів*

4. Предмет дослідження *Методи машинного навчання для класифікації текстових даних*

5. Перелік завдань, які потрібно розробити *розробка формальної постановки задачі; аналіз існуючих підходів до класифікації текстів; розробка алгоритму*

класифікації наукових статей за УДК; програмна реалізація алгоритму

6. Перелік графічного матеріалу

Розподіл навчальних текстів за УДК,

Схема роботи алгоритму класифікації

Діаграма класів розробленого класифікатора

7. Орієнтовний перелік публікацій *Аналіз методів класифікації наукових текстів, Класифікація наукових статей за допомогою методів машинного навчання*

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання “ *01* ” *вересня* 20 *19* р

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Огляд існуючих алгоритмів класифікації текстів*	*13.09.2019*	
2	*Постановка та формалізація математичної моделі задачі*	*27.09.2019*	
3	*Розробка інформаційного та програмного забезпечення*	*18.10.2019*	
4	*Проведення експериментальних досліджень розробленого алгоритму*	*01.11.2019*	
5	*Оптимізація параметрів машинного навчання*	*15.11.2019*	
7	*Оформлення документації*	*29.11.2019*	
8	*Подання роботи на попередній захист*	*05.12.2019*	
9	*Подання роботи на основний захист*	*16.12.2019*	

Студент _____ (підпис) _____ (ініціали, прізвище)

Науковий керівник _____ (підпис) _____ (ініціали, прізвище)

РЕФЕРАТ

Актуальність теми: для спрощення пошуку необхідної інформації серед наукових публікацій в Україні використовується бібліотечна класифікація. Проте наразі ця система є недосконалою, адже при класифікації допускаються помилки, а в деяких випадках вона виконується для збірника загалом, що призводить до часткової невідповідності для деяких статей, що в нього входять. Також виконання класифікації сторонньою людиною (наприклад, бібліотекарем чи редактором) вимагає багато часу. Вирішенням цієї проблеми є автоматизація процесу класифікації. За рахунок використання машинного навчання можна створити автоматичний класифікатор, яких дозволить покращити точність класифікації порівняно з ручною та прискорити класифікацію нових надходжень.

Мета дослідження: створення класифікатора наукових статей за категоріями УДК на основі машинного навчання.

Для реалізації поставленої мети були сформульовані **наступні завдання**:

– систематизація існуючих алгоритмів класифікації текстових даних;

– збір достатньої навчальних даних, розробка класифікатору на основі машинного навчання;

– тестування та аналіз ефективності отриманого алгоритму;

– визначення подальшого напрямку досліджень.

Об’єкт дослідження: бібліотечна класифікація наукових статей.

Предмет дослідження: алгоритми класифікації текстових даних.

Методи дослідження: для розв’язання поставленої задачі використовувались наївний баєсів класифікатор, нейронні мережі, алгоритм зворотного поширення помилки.

Наукова новизна: найбільш суттєвими науковими результатами магістерської дисертації є дослідження можливостей автоматизації класифікації наукових текстів; пошуку помилок у вже класифікованих текстах; створення алгоритмів класифікації для розрізнення категорій у текстів близьких тематик.

Практичне значення отриманих результатів визначається тим, що запропонований алгоритм дозволяє досягти точності бібліотечної класифікації в 86%, що дозволяє використовувати його для пошуку і виправлення помилок у класифікації текстів, а також як допоміжного засобу при класифікації нових надходжень.

Зв'язок роботи з науковими програмами, планами, темами: робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Математичні моделі та технології в СППР». Державний реєстраційний номер 0117U000914

Апробація: основні положення роботи доповідались і обговорювались на XII науково-практичній конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» (ПМК-2019), а також на третій всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019).

Ключові слова: класифікація текстів, нейронні мережі, УДК, машинне навчання.

ABSTRACT

Relevance: to simplify the search for relevant information among scientific publications in Ukraine, a library classification is used. However, this system is not perfect at this time, because classification is erroneous, and in some cases it is executed for the journal as a whole, which results in partial discrepancies for some of its articles. Also, it takes a long time to perform the classification by a third party (such as a librarian or editor). The solution to this problem is to automate the classification process. By using machine learning, automatic classifier can be created, which will improve the accuracy of the classification compared to manual and accelerate the classification of new revenues.

Purpose: create a classifier of scientific articles by UDC categories based on machine learning.

To achieve this goal, the following tasks were formulated:

- systematization of existing text data classification algorithms;
- gathering sufficient training data, developing a classifier based on machine learning;
- testing and analysis of the efficiency of the obtained algorithm;
- determining the further direction of research.

Object of study: library classification of scientific articles.

Subject of study: algorithms for classification of text data.

Research methods: naive Bayes classifier, neural networks, backpropagation algorithm were used to solve this problem.

Scientific novelty: the most significant scientific results of a master's thesis are the study of the possibilities of automation of the classification of scientific texts; search for mistakes in already classified texts; creation of classification algorithms for distinguishing categories in texts of similar subjects.

The practical value of the obtained results is determined by the fact that the proposed algorithm allows to achieve the accuracy of library classification in 86%, which allows to use it for finding and correcting errors in the classification of texts, as well as an aid in the classification of new receipts.

Relationship with working with scientific programs, plans, topics: work was performed at the Department of Automated Information Processing and Management Systems of the Igor Sikorsky National Technical University of Ukraine «Kyiv Polytechnic Institute» within the topic «Mathematical Models and Technologies in DSS». State Registration Number 0117U000914

Approbation: the main provisions of the work were reported and discussed at the XIII Scientific and Practical Conference of undergraduate and graduate students «Applied Mathematics and Computing» (AMP-2019), as well as at the third all-Ukrainian scientific and practical conference of young scientists and students «Information Systems and Technologies of Management» (ISTM-2019).

Keywords: text classification, neural networks, UDC, machine learning.

ЗМІСТ

ВСТУП

Одним з методів, що використовується для спрощення пошуку необхідної інформації, є бібліотечна класифікація. Є багато систем, які описують структуру каталогу, і одним з прикладів є універсальна десяткова класифікація, що застосовується в Україні. Проте наразі у нього є дві основні проблеми: по-перше, автори, що визначають УДК для своєї статті не завжди роблять це правильно, що ускладнює пошук інформації; по-друге, не для всіх публікацій вказується цей класифікатор.

Вирішенням цих проблем є введення автоматичної класифікації текстів, що дозволить виправити помилки, що були допущені при ручній класифікації та дасть можливість виконувати класифікацію для будь-яких текстів як тільки вони надходять до бібліотеки.

Відповідно, метою даної роботи є розробка математичного та програмного забезпечення для класифікації наукових статей за УДК за допомогою машинного навчання.

Для досягнення мети необхідно дослідити наявні алгоритми класифікації текстів, порівняти їх ефективність, проаналізувати та обрати найбільш оптимальні алгоритми присвоєння УДК науковим статтям, в тому числі для випадків коли стаття належить декільком категоріям. Крім того, необхідно створити прикладне програмне забезпечення для навчання класифікаторів та виконання класифікації тексту, введеного користувачем.

1 ПІДХОДИ ДО КЛАСИФІКАЦІЇ ДАНИХ

1.1 Постановка задачі класифікації

Класифікація текстів – одна з задач машинного навчання, що полягає у зарахуванні заданого документа до однієї або декількох категорій на основі його змісту. Математична постановка завдання виглядає наступним чином: для заданих множин текстів $T = \{t_1, t_2, \dots, t_n\}$, категорій $C = \{c_1, c_2, \dots, c_m\}$ та функції класифікації $\Phi: T \times C \to \{0, 1\}$ необхідно знайти функцію Φ', яка буде максимально близькою до цільової.

1.2 Загальні принципи класифікації даних

Класифікація текстів є однією з задач машинного навчання. Машинне навчання полягає в тому, що комп'ютер самостійно шукає закономірності в даних, для яких важко скласти певний визначений алгоритм. За допомогою спеціальних підходів комп'ютер аналізує вхідні дані та може знайти спільні риси у різних об'єктів, які людина не змогла б помітити через неймовірний об'єм таких даних.

Для класифікації текстів використовується навчання з учителем. Ця концепція близька до людського навчання. Вона полягає в тому, що на початку роботи програмі надаються тексти, для яких вже визначена категорія. Після аналізу (який власне називається навчанням) програма знаходить типові відмінності між категоріями та готова виконувати класифікацію реальних даних. Потенційною проблемою є те, що під час навчання будуть виділені ознаки, що специфічні для конкретних текстів, а не категорії в цілому, тому початкова множина текстів ділиться на дві частини – навчальну та тестову. Навчальна множина подається на вхід програми та на ній виконується навчання. Тестова множина використовується для перевірки ефективності навчання. Таким чином якщо програма буде запам'ятовувати специфічні, а не узагальнені ознаки, то на тестовій множині вона покаже результати, близькі до випадкових, адже жодного відомого тексту там не буде.

Ще однією проблемою є незбалансованість класів, для яких виконується класифікація. Наприклад при пошуку хворих людей здорових може бути переважаюча кількість (наприклад 90%), а тому програма, яка буде завжди казати що людина здорова незалежно від вхідних даних буде помилятись лише кожен десятий раз. Ця проблема буде відчутною у випадку якщо основною задачею програми є саме виявлення хворих людей. Тому для машинного навчання введено кілька інших метрик, які дозволяють всесторонньо оцінювати ефективність класифікації та уникати подібних проблем [1].

Для початку визначимо результати класифікації:

- TP (англ. true positives) – кількість релевантних елементів, що були розпізнані правильно. В прикладі з пошуком хвороби це кількість хворих людей, що були розпізнані як хворі;
- FN (англ. false negatives) – кількість нерелевантних елементів, що були розпізнані правильно. В прикладі з пошуком хвороби це кількість здорових людей, що були розпізнані як здорові;
- TN (англ. true negatives) – кількість релевантних елементів, що були розпізнані неправильно. В прикладі з пошуком хвороби це кількість хворих людей, що були розпізнані як здорові;
- FP (англ. false positives) – кількість нерелевантних елементів, що були розпізнані неправильно. В прикладі з пошуком хвороби це кількість здорових людей, що були розпізнані як хворі.

Очевидно що при діагностуванні хвороби ідеальним випадком є зведення FP до нуля, тому що таким чином ми не пропустимо жодного хворого. Але в такому випадку може виявитись що TN, тобто кількість людей, які здорові, але діагностовані як хворі, виявиться надто великою, що може значно погіршити реальну ефективність застування цього алгоритму. Тому на основі цих результатів обчислюються інтегральні метрики, за якими і оцінюють ефективність класифікації.

Інтуїтивною метрикою є відсоток правильних відповідей, що можна обчислити за формулою $accuracy = \frac{TP+FN}{TP+TN+FP+FN}$. Але, як було вказано раніше, така метрика не завжди показує реальну ефективність роботи класифікатора (особливо це видно на задачах з незбалансованими класами).

Тому в машинному навчанні використовуються дві додаткові метрики. Першою є точність, що обчислюється за формулою $precision = \frac{TP}{TP+FP}$. Це значення відповідає на питання «Яка кількість елементів є релевантною серед обраних?». В прикладі з пошуком хвороби це буде кількість реально хворих людей, серед тих, кого програма відправила на обстеження. Другою метрикою є повнота, що обчислюється за формулою $recall = \frac{TP}{TP+TN}$, що відповідає на питання «Яка кількість релевантних значень була успішно отримана?». У прикладі з пошуком хвороби це буде кількість людей, яких ми відправили на обстеження по відношенню до загальної кількості хворих. Крім того, як єдиний параметр оцінки якості роботи моделі використовується метрика F1 (також відома як F-міра або F-оцінка). Вона обчислюється за формулою $F_1 = (1 + \beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision)+recall}$. При значенні $\beta = 1$ вона перетворюється на середнє гармонійне параметрів $precision$ та $recall$: $F_1 = 2\frac{precision \cdot recall}{precision + recall}$. Проте підстановка замість β додатніх чисел дозволяє регулювати вплив параметрів $precision$ та $recall$ на загальний результат, і відповідно визначати що для класифікатора є важливішим – не упустити релевантні значення чи мінімізувати кількість отриманих нерелевантних (важливість знайти всіх хворих чи не відправляти на лікування здорових у нашому прикладі).

Тим не менш, простого вдосконалення метрик недостатньо. Дослідження показують, що для класифікації різних типів даних ефективними є різні алгоритми, тому при дослідженні ефективності класифікації важливо оцінити різні алгоритми класифікації та обрати серед них найкращий. Основними алгоритмами, що застосовуються для класифікації даних є [2]-[3]:

- наївний баєсів класифікатор;

- метод опорних векторів;
- метод найближчих сусідів;
- дерева прийняття рішень;
- нейронні мережі.

В свою чергу, нейронні мережі мають багато різновидів, кожен з яких поводить себе по іншому та підходить но конкретних типів даних. Серед них найбільш поширеними є наступні [4]:

- нейронні мережі прямого поширення (англ. Feedforward Neural Networks);
- згорткові нейронні мережі (англ. Convolutional Neural Networks);
- рекурентні нейронні мережі (англ. Recurrent Neural Networks);
- нейронні мережі, що використовують механізм «уваги» (англ. Attention-based Neural Networks).

Крім того, існує велика кількість нейронних мереж, що складаються з вище зазначених та інших мереж як із структурних блоків, наприклад, рекурентні згорткові нейронні мережі (англ. Recurrent Convolutional Neural Networks або RCNN) чи мережі з довгою короткотривалою пам’яттю (Long Short-Term Memory Neural Networks або LSTM NN). На відміну від класичних алгоритмів, що були створені наприкінці минулого століття, для нейронних мереж найбільш ефективним є модифікація структури під кожну конкретну задачу.

1.3 Представлення вхідних даних

У будь-яких текстів є певні елементи, які містяться там лише для спрощення сприйняття. Наприклад, до них відноситься більшість розділових знаків. Крім того, багато елементів тексту не є специфічними для певного класу, а тому лише ускладнюють класифікацію. Ними можуть бути деякі слова, наприклад займенники чи прийменники.

Крім того, проблемою при обробці текстів є різноманітність форм, яких можуть набувати слова. Ця проблема є менш актуальною для мов германської групи (англійська, німецька), на яких наразі виконується більшість досліджень,

в тому числі і з обробки текстів, проте в слов'янських мовах вона є відчутною. Наприклад, слова української мови можуть мати різний рід (наприклад «робив» та « робила») або різний час (наприклад «робив» та «робить»). Ці відмінності не є значущими з точки зору класифікації, і хоча їх можна обробити еквівалентним чином, проте такий підхід потребуватиме пропорційно більше обчислювальних ресурсів та тренувальних прикладів.

Тому для покращення якості класифікації перед передачею тексту на вхід до основного алгоритму тексти треба обробити. Під час цього відбувається видалення незначущих та малозначущих елементів, таких як знаки пунктуації або числа, що дозволяє значно пришвидшити процес навчання та спростити виділення важливих ознак тексту на основі яких відбувається класифікація, внаслідок чого покращується точність та впевненість класифікації [5].

Основними етапами перетворення тексту перед класифікацією є:

а) токенізація. На цьому етапі відбувається поділ тексту на елементи. Зазвичай такими елементами є слова, і розділення відбувається по символам пробілів;

б) усунення шуму. Більшість алгоритмів класифікації не оперує семантичними одиницями, більшими за слово. Тому поділ тексту на складові, наприклад речення, абзаци чи інші структурні елементи є непотрібними. В такому випадку на цьому етапі виконується видалення розділових знаків. Також на цьому етапі може відбуватись видалення інших символів у випадку якщо вони не є необхідними для класифікації, наприклад, слова на інших мовах або числа. У випадку якщо поділ на семантичні одиниці все ж таки потрібен при видаленні відповідних розділових знаків відбувається маркування;

в) видалення стоп-слів. Не всі слова є однаково корисними для класифікації. Наприклад, займенники не несуть додаткового змісту, а лише спрощують сприйняття тексту людиною. Тому для покращення ефективності алгоритму їх необхідно видаляти. На відміну від попередніх кроків, які відбуваються незалежно від тематики тексту (і, в деяких випадках, навіть його

мови) на цьому етапі зазвичай використовується спеціальний словник, в який заносяться слова, що необхідно видаляти;

г) стеммінг або лематизація. На цьому етапі відбувається конвертація слів у семантичні одиниці. Це найскладніший етап підготовки тексту до класифікації. Навіть на сьогоднішній день тривають дослідження та пошук оптимальних алгоритмів перетворення [6]-[8]. Стеммінг це процес зведення слова до його початкової форми. В процесі стеммнігу відкидаються морфологічні складові слів. В більшості випадків це лише закінчення. Проте іноді також відкидаються суфікси та/або префікси, що може призвести до погіршення якості класифікації, проте значно покращує швидкість навчання та класифікації. Лематизація є складнішим процесом. У ньому слова зводяться не своєї основи не тільки за допомогою узагальнених правил, а й з використанням спеціальних словників. Там зазначаються різні форми слів та словосполучень, що мають еквівалентний семантичний зміст. Такий підхід дозволяє точніше зберігати початковий зміст тексту, проте вимагає підготовки додаткових даних для роботи. Крім того, для більшості заданих словники необхідно оновлювати, адже значення слів та їх еквівалентність може мінятись в різних галузях.

Після цього отримані слова конвертують у потрібний для програми формат. Зазвичай для всієї множини створюють словник, у якому кожному слову присвоюється числовий ідентифікатор. В результаті цього програма отримує на вхід числа, з якими набагато простіше працювати. Крім того, такий підхід дозволяє скоротити використання пам’яті, адже в найбільш поширених ситуаціях одне число займає значно менше пам’яті ніж масив символів. Недоліком є неможливість для людини зрозуміти що мається на увазі у даному тексті. Для отримання формату, зрозумілого людині необхідно проводити зворотне перетворення на основі раніше створеного словника.

1.3.1 Модель Bag-of-Words

Після підготовки тексту для сприйняття програмою необхідно перетворити його в формат, що буде сприйнятий машиною. Першим форматом, який почав широко використовуватись у класифікації текстів став Bag-of-Words

[9]. За своєю суттю це таблиця, в який для кожного слова знаходиться кількість його входжень в даному тексті. Цей підхід вирішує основну проблему більшості алгоритмів, що працюють з класифікацією текстів: вони потребують однакової довжини для всіх текстів. При сприйнятті текстів як потоку слів цього неможливо досягти, адже всі тексти мають різну довжину. З даним підходом розмір вхідних даних строго фіксований – він рівний кількості елементів, які знаходяться у словнику, що був створений на етапі підготовки текстів. Проблемою даного підходу є те, що слова, які не зустрічались в тренувальних даних, не попадуть на вхід до класифікатора взагалі, адже вони відсутні в словнику. Проте зазвичай це не є проблемою, адже якщо класифікатор не навчався сприймати такі слова, то й отримати корисної інформації від них він не зможе. Крім того, при такому підході велика частка чисел буде рівна нулю, адже малоймовірно щоб текст використовував всі можливі слова.

1.3.2 Модель TF-IDF

Модель TF-IDF є покращенням Bag-of-Words. Вона покликана вирішити наступну проблему: якщо між текстами є велика різниця в довжині, то навіть мало спеціалізований текст може містити слова з різних галузей. Дослідження показали, що набагато важливіше мати відносну частоту входжень слова в текст, ніж абсолютне число, що позначає загальну кількість входжень [10]. Відповідно, чисельне представлення важливості слова обчислюється за наступними формулами:

$$tf_i = \frac{w_i}{\sum_k w_k}, k = 0..n$$

де n – загальна кількість унікальних слів у документі,

w_i – кількість i-х слів у документі.

Ця формула використовується для визначення частки входження слова в документі.

$$idf_i = \frac{|D|}{|d \supset t_i|},$$

де $|D|$ – загальна кількість документів,

$|d \supset t_i|$ – кількість документів, в яких зустрічається слово t_i.

Ця формула покликана зменшити вагу загальновживаних слів.

Загальне представлення слова на вході до класифікатора обчислюється наступним чином:

$$input_i = tf_i \cdot idf_i$$

Таким чином ранг слова підвищується, якщо воно часто зустрічається в заданому тексті та зменшується якщо це слово часто зустрічається в інших текстах. Це призводить до того, що спеціалізовані терміни, що часто зустрічаються лише в профільних текстах та не зустрічаються в інших, оцінюються набагато вище ніж слова загального вжитку, що зустрічаються потроху в усіх текстах.

1.3.3 Вектор термів

У розглянутих вище прикладах основною проблемою є страта контексту. Ми знаємо, що слово зустрілося кілька разів у тексті, але де саме? Його входження могли бути згруповані в одному абзаці, а могли бути рівномірно розташовані серед усіх інших. Деякі слова також можуть мати різні значення в залежності від контексту. Тому використання множини слів замість їх послідовності зменшує кількість інформації, яку отримує класифікатор, а тому погіршує точність класифікації.

Для уникнення цієї проблеми на вхід подають послідовність чисел, що позначають слова, в тому ж порядку, в якому вони зустрічаються в оригінальному тексті. Це дає можливість аналізувати не тільки слово саме по собі, але і його сусідів. Основною проблемою такого підходу є те, що більшість алгоритмів що використовуються для класифікації текстів не здатні обробляти вхідні дані змінної довжини, а тому такий підхід отримав популярність тільки з розвитком рекурентних нейронних мереж, які, по суті, є єдиним алгоритмом, що здатний обробляти послідовність термів незалежно від їх довжини.

1.3.4 Використання нейронної мережі Word2vec

Із використанням контексту для класифікації текстів задача ускладнюється. По суті задача починає поділятись на дві – аналіз тексту для

отримання значень слів та власне класифікація на основі отриманої семантики. Для вирішення першої частини цієї задачі компанія Google в 2013 році створила нейронну мережу для отримання семантики слів. Вона широко використовується для отримання контексту слів при роботі з текстами. Після відповідного навчання вона здатна перетворювати вхідні слова на вектори в спеціальному просторі. При цьому векторний добуток двох представлень буде відповідати їх реальній близькості в семантиці текстів. Це дозволяє набагато точніше обробляти синоніми та інші слова, що схожі за своїм значенням набагато простіше, що в свою чергу підвищує точність класифікації. Основним недоліком цієї мережі є те, що вона розроблювалась приватною компанією і тільки для англійської мови. І навіть враховуючи те, що структура мережі опублікована, повторити отриманий результат на іншій мові практично неможливо, адже навчання такої мережі потребує неймовірної кількості обчислювальних ресурсів та текстів для навчання цієї мережі. Наразі більшість опублікованих досліджень використовують тексти на англійській мові, тому вони без проблем користуються даною мережею для аналізу, проте для української мови необхідно шукати інший підхід для отримання семантики слів.

1.4 Алгоритми класифікації

1.4.1 Наївний баєсів класифікатор

Наївний баєсів класифікатор є одним з найперших методів класифікації даних. Цей алгоритм є простим та потужним, а тому використовується для класифікації даних і на сьогоднішній день. Його перевагами є простота, висока швидкість навчання та класифікації, низьке використання пам’яті, а також невеликий об’єм навчальних даних, при яких він починає ефективно працювати. Крім того, при появі додаткових даних його можна просто та швидко оновити. Також до його переваг можна віднести універсальність – на відміну від нейронних мереж або інших алгоритмів, які спеціалізуються лише на даних певного типу, баєсів класифікатор чудово працює в будь-якій ситуації.

Він базується на теоремі Баєса, яка оцінює ймовірність настання події за її ознаками. Відповідно до цієї теореми, ймовірність належності тексту T з вектором ознак $x = \{x_1, x_2, \dots, x_n\}$ до класу C_k оцінюється за наступною формулою:

$$p(C_k \mid x) = \frac{p(C_k) \cdot p(x \mid C_k)}{p(x)}.$$

Члени $p(C_k)$ та $p(x)$ можна обчислити при підготовці даних, це ймовірність зустріти певну ознаку та категорію відповідно. Тому основний зміст при класифікації текстів має саме $p(x \mid C_k)$. Оскільки «наївність» даного класифікатора полягає в тому, що всі ознаки вважаються незалежними, то цей множник можна перетворити наступним чином: $p(x \mid C_k) = \prod_{i=0}^{n} p(x_i \mid C_k)$.

Зі зростанням кількості інформації, що зберігалася на електронних носіях почалися перші спроби автоматичної категоризації та класифікації. Наприкінці минулого століття було створено перші класифікатори текстів, що використовували баєсів класифікатор у якості основного алгоритму. В дослідженні [11] використання баєсова класифікатора показало точність 74% при використанні словника з 10000 слів. Важливою деталлю дослідження є те, що при зростанні величини словника понад оптимальних розмір зменшується точність класифікації. Так, при збільшенні словника до 20000 слів точність класифікації падає на два пункти до 72%.

На сьогоднішній день, цей алгоритм використовується відчутно рідше, адже зі зростанням потужності комп'ютерів одна з його основних переваг – низька обчислювальна складність перестала так відчутно впливати на вибір. В сучасних дослідженнях основним застосуванням баєсового класифікатора є використання його як точки відліку для порівняння ефективності нових алгоритмів.

1.4.2 Метод опорних векторів

Наївний баєсів класифікатор має відчутний недолік – при його застосуванні вважається що всі параметри системи є незалежними. Проте у

випадку для класифікації текстів це твердження не є правдивим. Слова у текстах з’являються не у випадковому порядку, наявність синтаксичних конструкцій та загальновживана структура речень може з великою ймовірністю гарантувати появу одних слів після інших.

Метод опорних векторів позбавлений даного недоліку. Він полягає в поділі простору вхідних даних на частини, що відповідають заданим класам. При цьому в процесі навчання основною задачею є мінімізація точок, що потрапляють в чужу категорію та максимізація вільного простору між категоріями. На конкретних задачах такий підхід показує кращі результати, ніж баєсів класифікатор, проте він також має свої недоліки. Основним недоліком є необхідність правильно обрати функцію ядра. Навіть для схожих задач оптимальними можуть виявитись різні функції, тому ефективність застосування алгоритму залежить від дослідника. Також цей алгоритм призначений лише для бінарного поділу, тому для задач класифікації що мають більше ніж дві категорії він потребує додаткових перетворень для зведення такої задачі до бінарної класифікації.

1.4.3 Метод найближчих сусідів

Метод найближчих сусідів (також відомий як метод k-найближчих сусідів) полягає в пошуку вже класифікованих об’єктів, що є найближчими до заданого. Основною задачею при реалізації цього алгоритму є вибір правильної метрики для обчислення відстані між об’єктами. Найбільш поширеним є використання евклідової відстані, проте можливі і інші варіанти, наприклад манхеттенська відстань або відстань Хеммінга. Також важливим є вибір параметра k, який визначає кількість сусідів, яких необхідно знайти для визначення класу. При його збільшенні покращується точність класифікації, проте це також значно збільшує обчислювальну складність.

Перевагами даного методу є можливість класифікації на декілька категорій, що є перевагою порівняно з методом опорних векторів. Також методу не потрібні додаткові умови для вхідних даних (наприклад, вимога до незалежності параметрів, яка висувається в наївному баєсовому класифікаторі).

Також він достатньо простим як для розуміння, так і для реалізації та не потребує попереднього навчання. Крім того в нього можна легко додати нові навчальні приклади навіть в процесі роботи. Недоліками цього алгоритму є його ефективність тільки на маленьких наборах даних. Із ростом вхідних прикладів обчислювальна складність збільшується квадратично. Також такий алгоритм погано працює на наборах даних, що мають велику кількість вхідних параметрів. Функції обчислення відстані, наприклад евклідова чи манхеттенська, не вміють надавати пріоритет певним параметрам, а тому вони є неефективними, коли вхідні дані є негомогенними. Особливо важливим це є для текстів, тому що там завжди є загально вживані слова, які не вносять додаткового змісту для класифікації. При використанні даного методу вони будуть лише заважати, а тому текст необхідно буде додатково очищати від таких слів. Також він не має ніякої можливості компенсувати незбалансованість тренувальних даних, коли одна категорія зустрічається значно частіше ніж інша.

1.4.4 Нейронні мережі

Наразі нейронні мережі є найпотужнішим засобом для розв’язання більшості задач машинного навчання. Доведено, що за допомогою нейронних мереж можна нескінченно точно описати будь-яку функцію [12]. Хоча дослідження нейронних мереж почалося ще в середині минулого століття (вперше подібна структура була запропонована в 1943 році в роботі [13]), проте їх практичне застосування стало можливим відносно недавно. На це є дві причини. Першою є те, що нейронні мережі потребують значних обчислювальних ресурсів для своєї роботи (і особливо навчання). Лише недавно обчислювальні потужності комп’ютерів зросли до рівня, який дозволив виконувати навчання складних мереж у прийнятні строки (до декількох днів для складних мереж). Також причиною зростання ефективності нейронних мереж є зростання кількості даних що зберігаються електронно. Оскільки для свого навчання вони потребують неймовірної кількості даних, то навіть десятиліття тому їх навчання було доступне лише корпораціям які здатні виділити величезні ресурси для збору навчальних прикладів для таких мереж. Проте останніми

роками кількість відкритих даних постійно збільшується, а вартість обчислень зменшується, що дозволяє використовувати мережі всім бажаючим, і, відповідно, стимулює додаткові дослідження та покращення щодо їх роботи.

1.4.4.1 Нейронні мережі прямого розповсюдження

Найпростішою та першою винайденою структурою нейронної мережі є мережа прямого розповсюдження. В ній нейрони розташовані послідовними шарами, і інформація від кожного нейрону поточного шару передається кожному нейрону наступного шару. При цьому перший шар нейронної мережі називають вхідним, останній вихідним, а проміжні – прихованими. Приклад нейронної мережі з двома прихованими шарами зображено на рисунку 1.1.

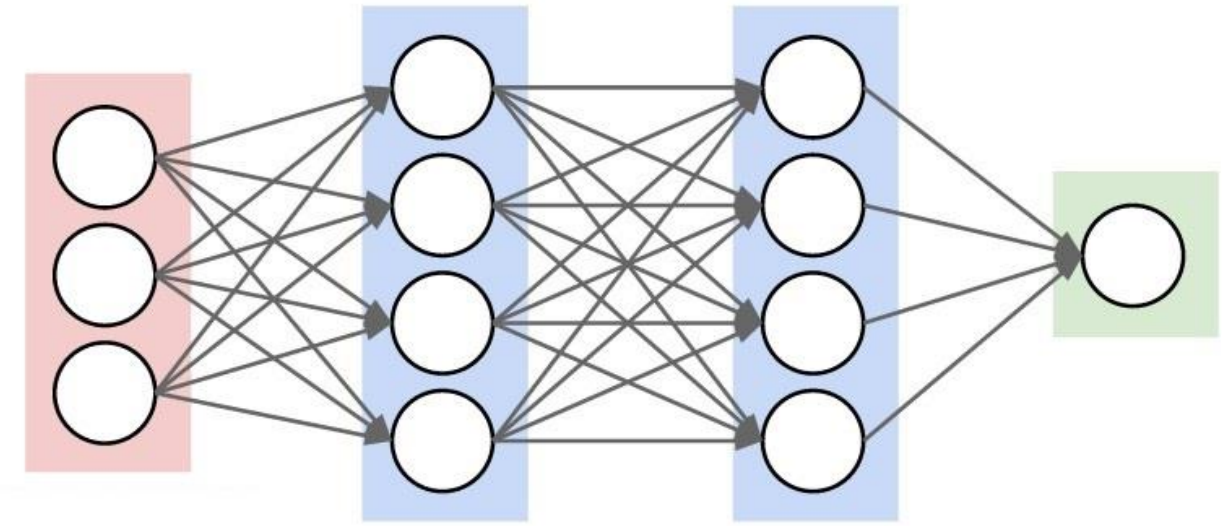

Рисунок 1.1 – Структура мережі прямого розповсюдження з двома прихованими шарами

При цьому вихід кожного нейрона представляю собою лінійну комбінацію вхідних параметрів:

$$p_j = \sum_i o_i \, w_{ij},$$

де p_j це вихід j-го нейрону поточного шару,

o_i – вихід i-го нейрону попереднього шару,

w_{ij} – лінійних коефіцієнт, що відповідає i-му сходу в поточний нейрон.

Відповідно задачею навчання нейронної мережі є підбір коефіцієнтів w_{ij} таким чином, щоб отримати результат, найближчий до бажаного. Значно кращих результатів роботи нейронної мережі можна досягти якщо використовувати додаткову функцію, яка буде перетворювати вихід від нейрона таким чином, щоб він потрапляв на проміжок $[0, 1]$ [14]. Наразі, в якості таких функцій використовують арктангенс, сигмоїдальну функцію або ReLU (англ. rectified linear unit).

Для навчання нейронних мереж використовується алгоритм зворотного поширення помилки. По своїй суті це застосування алгоритму Гауса–Ньютона для мінімізації функції відхилення очікуваного результату від бажаного. На кожній ітерації цього алгоритму виконується обчислення відповіді нейронної мережі для заданого тренувального прикладу. Далі виконується обчислення функції похибки як середньоквадратичного відхилення отриманого значення від бажаного. Після цього починається зворотне розповсюдження помилки: для кожного шару, починаючи з останнього, обчислюється градієнт та відбувається модифікація коефіцієнтів функції в протилежному до градієнта напрямку. Далі аналогічна похибка обчислюється для наступного шару нейронної мережі.

1.4.4.2 Згорткові нейронні мережі

Проблемою звичайних нейронних мереж прямого розповсюдження є швидке зростання обчислювальної складності при зростанні кількості нейронів, адже кожен нейрон зв’язаний з один з одним. Проте не завжди значення з віддалених частин вхідних даних реально впливають одне на одного. Наприклад в текстах контекст важливий лише для слів, що стоять поряд. Слова, що розташовані на відстані кількох абзаців зазвичай не мають ніякого семантичного відношення між собою. Тому для спрощення обчислювальної складності було створено згорткові нейронні мережі. В оригінальному дослідженні [15] їх пропонувалось використовувати лише для обробки зображень, проте вони показують свою високу ефективність і в інших галузях [16].

Основним елементом згорткових нейронних мереж, порівняно з іншими типами мереж, є спеціальний тип нейронного шару – згортка. На відміну від повнозв'язних шарів, на згортковому шарі кожен нейрон обробляє лише обмежену кількість входів – своє рецептивне поле. За рахунок цього значно зменшується кількість зв'язків нейронної мережі, і, відповідно, її обчислювальна складність. Тим не менш, в загальному випадку такі мережі мають меншу точність порівняно зі співставними по кількості шарів повнозв'язними нейронними мережами, адже аналізуються зв'язки не між всіма елементами. Тому важливою задачею при розробці таких мереж є визначення необхідної величини рецептивного поля, яке повинно бути достатньо великим щоб не втрачати оточуючий контекст, та достатньо маленьким, щоб давати приріст продуктивності. Тим не менш, при правильній конфігурації такі мережі показують більшу ефективність, адже за рахунок вивільнення ресурсів на початку аналізу можна створювати більш складні нейронні структури при аналізі виділених ознак.

1.4.4.3 Рекурентні нейронні мережі

Розглянуті вище структури нейронних мереж мають спільний недолік – вони пристосовані тільки для обробки вхідних даних фіксованого розміру. Для деяких типів даних такий підхід є прийнятним, адже їх можна привести до такої форми – наприклад графічні зображення можна стиснути чи розтягнути. Проте для роботи з текстами необхідно використати інший підхід, адже в них на етапі попередньої обробки вхідних даних неможливо визначити які слова є важливими, а які ні.

На відміну від нейронних мереж прямого поширення рекурентні нейронні мережі мають додаткові зв'язки між нейронами між останнім та першим прихованим шаром, а граф зв'язків стає орієнтованим у часі. Схематична структура такої мережі зображена на рисунку Рисунок 1.2.

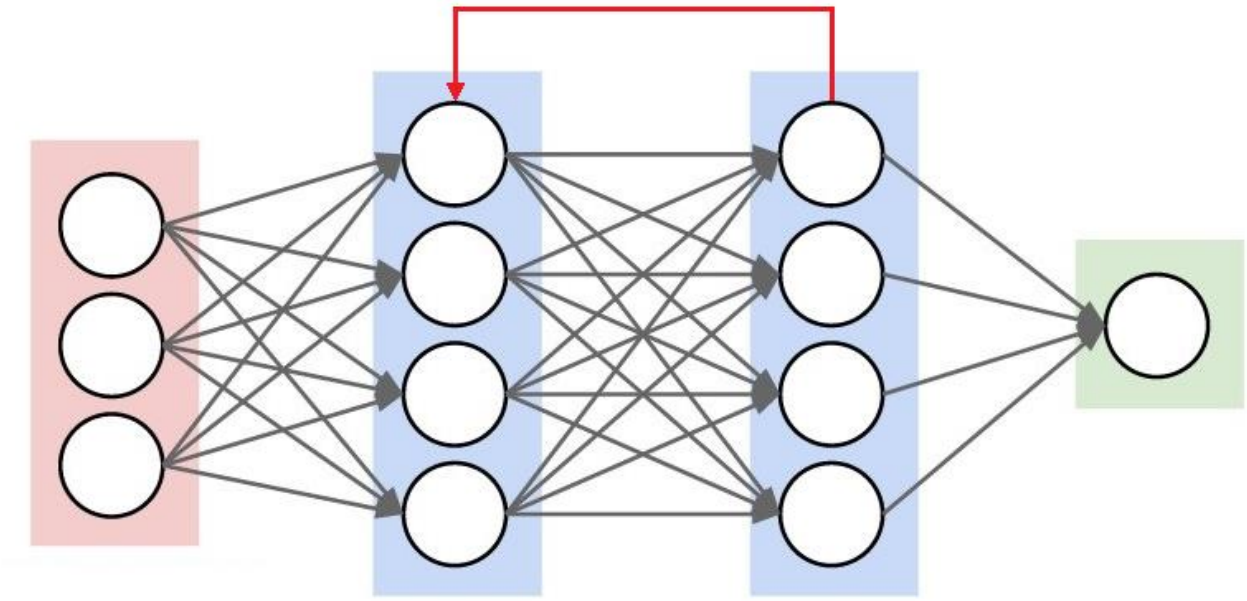

Рисунок 1.2 – Структура рекурентної нейронної мережі

В результаті нейрона мережа отримує можливість обробити кілька векторів вхідних даних використовуючи додаткові зв’язки для збереження даних, отриманих від обробки попередніх вхідних даних. Таким чином нейронна мережа отримує деяку подобу постійної пам’яті, що дозволяє обчислювати остаточний результат на основі потоку вхідних даних. Такі мережі можуть використовуватись як для обробки скінчених даних, які мають невизначену довжину, наприклад тексти, так і для обробки нескінчених даних, наприклад відео чи аудіо в реальному часі. В результаті застосування цього способу замість використання множини слів, в якій втрачається їх послідовність, і, відповідно, додаткова інформація, яка могла допомогти розпізнаванню, з’являється можливість обробляти послідовність слів цілком, що може покращити точність розпізнавання.

1.4.4.4 Нейронні мережі з довготривалою пам’яттю

Нейронні мережі з довготривалою пам’яттю є подальшим розвитком рекурентних мереж. Вони показують значно вищу ефективність та утримують перше місце в розв’язанні багатьох задач, в тому числі розпізнавання мови та рукописного тексту. Додатковим блоком, що використовується в таких мережах є спеціальний блок пам’яті. За структурою він аналогічний звичайному

рекурентному блоку, проте з однією відмінністю – для його нейронів не застосовується функція активації. Внаслідок цього значення, що зберігається всередині є незмінним в часі, а також штраф, який обчислюється при використанні метода зворотного поширення помилки не слабшає.

1.5 Висновки та постановка задач дослідження

Основною відмінністю розглянутих робіт є використання текстів, написаних англійською мовою. Навіть у вітчизняних дослідженнях (наприклад [17]) використовується англійська мова, оскільки для неї є значно більше досліджених та розроблених інструментів класифікації. Крім того, розглянуті роботи використовують великі корпуси простих текстів (наприклад, відгуки, коментарі чи повідомлення в Twitter), а класифікація великих текстів із використанням порівняно малої кількості навчальних даних є практично недослідженою. Отже необхідно дослідити ефективність застосування різних методів класифікації для великих текстів українською мовою. Для досягнення поставленої мети необхідно вирішити такі задачі: зібрати корпус текстів для навчання та тестування класифікаторів, виконати порівняльне дослідження ефективності роботи різних класифікаторів, знайти оптимальну структуру нейронної мережі для класифікації україномовних текстів (з урахуванням малого об’єму навчальних даних) та порівняти її ефективність з іншими методами класифікації.

2 АЛГОРИТМИ КЛАСИФІКАЦІЇ ТЕКСТІВ

2.1 Наївний баєсів класифікатор

В даній роботі досліджується ефективність класифікації текстів за допомогою нейронних мереж. Проте для оцінки їх результатів необхідно мати точку відліку, відносно якої будуть оцінювати отримані результати. Цей метод повинен мати доведену ефективність у подібних задачах та не містити алгоритмів, що специфічні для певної мови (наприклад англійської). Цим умовам відповідає наївний баєсів класифікатор. Він є одним з найперших алгоритмів, які використовувались для класифікації даних та показує високі результати при низьких вимогах до обчислювальних потужностей та навчальних прикладів.

Він базується на теоремі Баєса, яка оцінює ймовірність настання події за її ознаками. Відповідно до цієї теореми, ймовірність належності тексту T з вектором ознак $x = \{x_1, x_2, \dots, x_n\}$ до класу C_k оцінюється за наступною формулою:

$$p(C_k \mid x) = \frac{p(C_k) \cdot p(x \mid C_k)}{p(x)}.$$

Члени $p(C_k)$ та $p(x)$ можна обчислити на етапі підготовки даних, це ймовірність зустріти певну ознаку та категорію відповідно. Тому основний зміст при класифікації текстів має саме $p(x \mid C_k)$. Оскільки «наївність» даного класифікатора полягає в тому, що всі ознаки вважаються незалежними, то цей множник можна перетворити наступним чином: $p(x \mid C_k) = \prod_{i=0}^{n} p(x_i \mid C_k)$. Оскільки нам необхідно оцінити не абсолютну ймовірність, а тільки знайти найбільшу, то множником $p(x)$ можна знехтувати, адже він буде однаковим для кожної категорії. Відповідно, після всіх перетворень ми отримуємо остаточну формулу:

$$p(C_k \mid x) = p(C_k) * \prod_{i=0}^{n} p(x_i \mid C_k).$$

Алгоритм навчання класифікатора виглядає наступним чином: для кожного вектору ознак $X_i \in T;\ X_i = \{x_1, x_2, \dots, x_n\}$ у навчальних даних обчислюються наступні значення:

а) обчислюється кількість входжень заданих слів у текстах категорії, до якої належить поточний текст: $p_c := p_c + X_i$;

б) інкрементується кількість текстів, що належать до даної категорії: $count_c := count_c + 1;\ count_{total} := count_{total} + 1$;

в) Після обробки всіх текстів ми обчислюємо остаточні коефіцієнти для класифікації:

1) ймовірність зустріти текст в категорії C_k: $p(C_k) = \frac{count_c}{count_{total}}$;

2) ймовірність зустріти слово x_i в категорії C_k:

$$p(x_i \mid C_k) = \begin{cases} \frac{1}{count_c}, x_i = 0; \\ \frac{x_i}{count_c}, x_i > 0. \end{cases}$$

В формулі ймовірності використовується ненульова кількість входжень для того, щоб не виключати з розгляду тексти, які містять слова, що не зустрічались на етапі навчання. Таким чином, ці слова хоча й будуть значно зменшувати ймовірність потрапляння до цієї категорії, проте не будуть зменшувати її до нуля.

Під час класифікації необхідно перемножити отримані коефіцієнти та обрати максимальне значення ймовірності серед отриманих. Хоча такий підхід і прямо випливає з теоретичних розрахунків, проте він погано працює на практиці і потребує додаткового вдосконалення. При простому перенесенні математичних абстракцій в програмний код виникає проблема, пов’язана з тим, що комп’ютери оперують зі скінченою точністю, а тому перемноження малих чисел відбувається з відчутними втратами. Через це більш оптимальним вибором є перемноження логарифмів отриманих значень: оскільки логарифм є монотонно зростаючою функцією, а нам необхідно оцінити відносний пріоритет категорії, то з математичної точки зору такий підхід є еквівалентним, проте за рахунок

використання такого прийому збільшуються перемножувані числа і, відповідно, зменшуються втрати точності при множенні.

Перевагами даного алгоритму є можливість виконати найбільш затратну частину обчислень на етапі навчання і при класифікації, по суті, виконати лише один векторний добуток. Крім того, незважаючи на свою простоту, цей алгоритм показує високу точність класифікації та в більшості випадків працює краще ніж інші неспеціалізовані алгоритми.

2.2 Нейронні мережі

Наразі, провідним засобом для обробки інформації є нейронні мережі. Вони показують значно вищі результати ніж інші алгоритми машинного навчання практично в усіх задачах: ідентифікація та розмітка зображень, розпізнавання рукописного тексту та мови, в тому числі і в реальному часі, обробка текстів та машинний переклад, прийняття рішень, пошук закономірностей та класифікація даних тощо.

Нейронні мережі не можна запрограмувати в традиційному сенсі створення алгоритмів. Натомість, вона «навчається», тобто автоматично корегує коефіцієнти між нейронами для досягнення найкращого результату. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними та вихідними даними, а також виконувати узагальнення. За рахунок узагальнення нейронна мережа може працювати з вхідними даними, що були відсутні при навчанні, а також правильно обробляти зашумлені та частково викривлені дані. В результаті за рахунок своєї простоти та точності нейронні мережі стали провідним засобом для обробки інформації.

Математичною моделлю нейронних мереж є дерево матричних обчислень, що застосовується до вектору вхідних даних. Вихідне значення нейрону є лінійною комбінацією вектору вхідних даних X та вектору ваг W:

$$y = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = W^T X$$

Такий підхід використовувався досить довго через свою простоту та низьку обчислювальну складність, проте з розвитком задач, до яких

застосовувались нейронні мережі все більше і більше заважала проблема нелінійності реальних даних. Навіть при використанні декількох шарів, вихід нейронної мережі є лінійною комбінацією вхідних даних та матриці коефіцієнтів:

$$u = Ax + a;$$

$$y = Bu + b = B(Ax + a) + b = BAx + (Ba + b);$$

що еквівалентно

$$y = Cx + c; C = BA;\ c = (Ba + b)$$

Відповідно у цьому випадку виникає необхідність введення нелінійності до виходу нейронів, щоб вони могли наближувати відповідні дані. Таким чином, обчислення значення кожного нейрона виглядає наступним чином:

$$z = W^T X;$$

$$y = activation(z);$$

Різні функції дають різний тип нелінійності, а тому універсальної функції не існує і для кожного випадку необхідно шукати найбільш оптимальну.

2.2.1 Навчання нейронної мережі

Навчання нейронних мереж полягає в підборі коефіцієнтів матриці ваг W. Оригінальним алгоритмом навчання був алгоритм Гебба, що по суті відповідав стохастичному навчанню. Через його низьку ефективність створення ефективної нейронної мережі, яка могла б конкурувати з іншими алгоритмами, було дуже складним, а тому використання нейронних мереж було цікавим лише в дослідницьких цілях.

Ключовим кроком у розвитку нейронних мереж стала розробка алгоритму зворотного поширення помилки [18]. Він дозволяє цілеспрямовано корегувати значення нейронів щоб отримати бажане значення. Таким чином, алгоритм навчання нейронної мережі виглядає наступним чином:

а) на вхід нейронної мережі подається вектор значень X, що репрезентує навчальний приклад;

б) для кожного внутрішнього шару, починаючи від першого і до останнього:

1) кожен нейрон поточного шару обчислює зважену суму вхідних сигналів $z = W^T X$;

2) до отриманої функції застосовується функція активації $y = f(z)$;

3) із отриманих значень формується новий вектор, який використовується як вхідні дані для наступного шару;

4) значення, отримані на останньому шарі інтерпретуються як відповідь нейронної мережі;

в) далі для кожного шару нейронної мережі, починаючи від останнього до першого, відбувається корегування вагових коефіцієнтів;

1) для кожного нейрону поточного шару обчислюється помилка як $d = (target - actual) * f'(input)$;

2) зміна вагового коефіцієнта нейрону обчислюється як $\delta w = w + learning\_rate * d$;

3) до попереднього шару нейрону надсилається значення d у якості помилки.

Даний алгоритм дозволяє цілеспрямовано корегувати значення вагових коефіцієнтів нейронів для досягнення бажаного результату, що призвело до значного поширення нейронних мереж. За своєю суттю він є модифікацією градієнтного спуску, а тому при достатньо малому кроці він може знайти локальний мінімум функції втрат з будь-якою точністю.

Важливим параметром даного алгоритму є швидкість навчання, яка є множником при розповсюдженні помилки. Він визначає, на яку величину зміняться вагові коефіцієнти нейронів при помилковій відповіді. Для покращення точності роботи алгоритму бажано зробити цей коефіцієнт як можна меншим. Проте його зменшення пропорційно збільшує кількість кроків для досягнення мінімуму, і, відповідно збільшує час навчання. Тому для вирішення

реальних задач необхідно підбирати значення цього коефіцієнту для збереження балансу між точністю нейронної мережі та швидкістю її навчання.

2.2.2 Функції активації

Як було зазначено вище, використання виключно матричних операцій призводить до отримання лінійної залежності, при будь-який комбінації таких операцій. Відповідно для створення можливості обробляти нелінійні залежності між даними до нейронних мереж були додані функції активації. Вони застосовуються до виходу нейрона перед подачею сигналу до наступного шару, і, відповідно, формула для обчислення вихідного значення нейрону починає виглядати наступним чином:

$$z = W^T X; y = f(z)$$

Завдяки цьому, нейронні мережі мають можливість оцінювати не лише лінійні залежності між даними, а й виділяти інші типи закономірностей. Функцій активації є велика кількість, і кожна з них може використовуватись лише для певної задачі. Кожна з них відповідає окремому виду нелінійності, а загальний результат може досягатись як використанням однієї функції для всієї мережі, так і комбінуванням різних функцій.

Єдиним обмеженням для активаційних функцій є диференційованість, адже без цієї ознаки до отриманої нейронної мережі неможливо буде застосувати алгоритм зворотного поширення помилки.

Основними активаційними функціями, що використовуються в нейронних мережах прямого поширення є:

- логістична функція;
- гіперболічний тангенс;
- випрямляюча лінійна функція;
- експоненційна лінійна функція [19].

Розглянемо їх детальніше.

2.2.2.1 Логістична функція

Логістична функція, також відома як сигмоїда, або м’який крок, є найпоширенішою та найбільш пристосованою до загального використання функцією. Вона визначена наступним чином:

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Графік функції виглядає наступним чином:

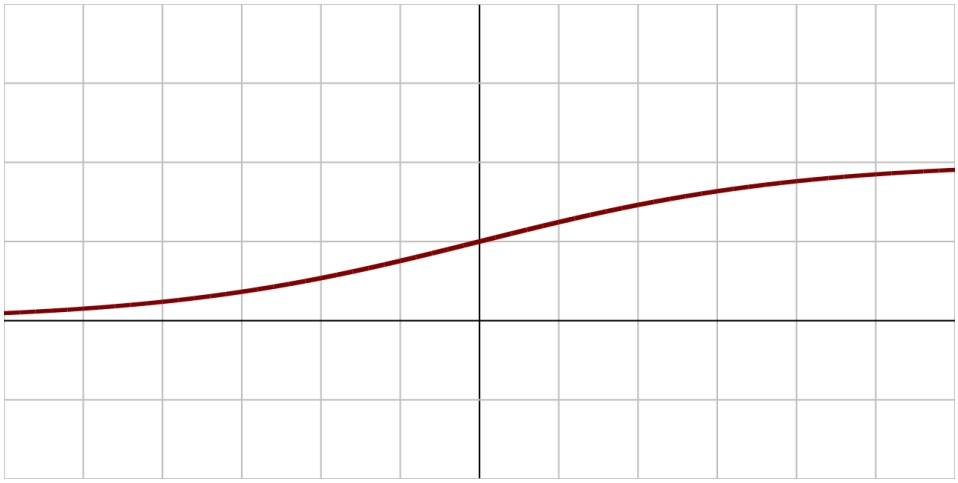

Рисунок 2.1 – Логістична функція

Перевагами такої функції є її чудова пристосованість для активації, де її кінцівки однозначно позначають чи належать вхідні дані до відповідного класу чи ні, а центр функції показує невпевненість нейронної мережі у результаті. Також вона видає значення від -1 до 1, що призводить до однорідних значень протягом всіх нейронних шарів.

Недоліком цієї функції є низьке значення похідної (градієнту), особливо для великих вхідних даних, що ускладнює навчання для таких випадків.

2.2.2.2 Гіперболічний тангенс

Гіперболічний тангенс має аналогічну до логістичної функції структуру. Його значення обчислюються за наступною формулою:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Як і сигмоїда, він приводить вхідні значення до вузького проміжку, що згладжує значення у випадку якщо певний нейрон має надто велике значення.

Проте на відміну від логістичної функції він є більш різким біля нуля, що чудово видно на графіку, особливо при прямому порівнянні:

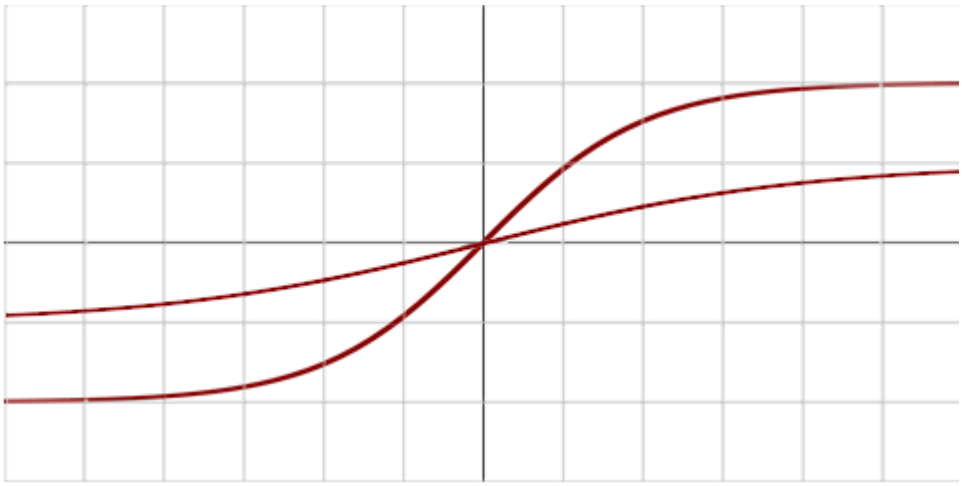

Рисунок 2.2 – Порівняння гіперболічного тангенсу та логістичної функції

Внаслідок цього, нейронна мережа може набагато простіше та точніше оперувати значеннями біля нуля. Крім того, на відміну від логістичної функції, гіперболічний тангенс зводить значення до проміжку (-1; 1), а значить від’ємні числа будуть строго від’ємними, а додатні строго додатними.

2.2.2.3 ReLU

Попередньо розглянуті функції вимагають значних обчислювальних потужностей як при обчисленні значення нейронної мережі при прямому проході, так і для обчислення значення похідної при зворотному проході. Там не менш, існують прості задачі, для яких подібні складні функції не є потрібними. Для спрощення обчислень у таких випадках використовується випрямляюча лінійна функція (англ. rectified linear unit або ReLU). Вона відтинає від’ємні значення на виході із нейрону:

$$ReLU(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$$

Її графік виглядає наступним чином:

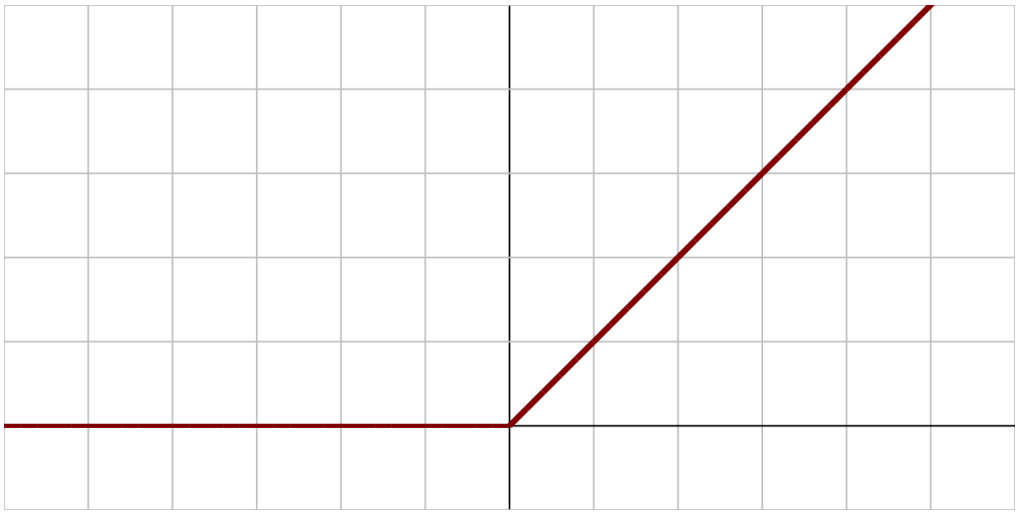

Рисунок 2.3 – ReLU

Також перевагою даної функції вирішення проблеми зникання градієнту, адже для попередніх функцій при виконанні зворотного поширення помилки кожен наступний шар отримує щоразу менші значення (які падають експоненційно), а тому при навчання багатошарових мереж відбувається дуже повільно через те, що вагові коефіцієнти на передніх шарах нейронів щоразу змінюються на мізерну величину. Через те, що для даної функції похідною є константа ‘1’, вона не зменшує значення помилки при її розповсюдженні до передніх нейронів, а тому навчання не сповільнюється.

2.2.2.4 ELU

Наступним еволюційним кроком стало використання експоненційної лінійної функції. Попередній варіант має недолік, що стає відчутним на великих мережах – так звані «мертві» нейрони. У випадку якщо значення зваженої суми є від’ємним, то вихід нейрона перетворюється на нуль. Відповідно похідна від нуля теж буде нуль, а тому при виконанні зворотного розповсюдження помилки зміна ваги нейрона домножується на похідну – нуль, а тому значення не змінюється взагалі. Відповідно, чим довше триває навчання, тим більше стає таких незмінних нейронів. Для уникнення цієї ситуації на від’ємній частині значень необхідно видавати близьке до нуля, проте не константне значення. В таких випадках використовується експоненційна лінійна функція, яка визначена наступним чином:

$$ELU(x) = \begin{cases} x, x > 0 \\ e^x - 1, x \leq 0 \end{cases}$$

Як видно із графіка, при вхідних числах, менших за нуль функція видає близькі до нуля значення, проте за рахунок використання експоненти у від'ємний половині області визначення, похідна такої функції не рівна нулю.

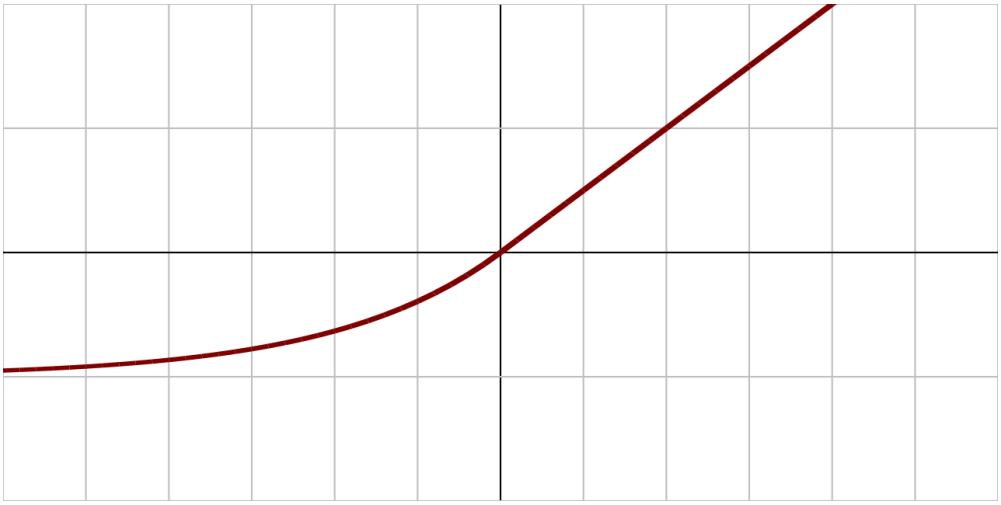

Рисунок 2.4 – ELU

2.2.3 Обчислення початкових коефіцієнтів

За замовчанням вагові коефіцієнти нейронів заповнюються нулями, і вже потім поступово отримують необхідні значення. Проте такий підхід є не надто ефективним, адже в великої кількісті стандартних задач початкові значення є приблизно відомими, і тому перші цикли навчання витрачаються на те, щоб дійти до цих значень. Щоб прискорити початкове навчання використовуються алгоритми попередньої ініціалізації вагових коефіцієнтів нейронів. В деяких випадках встановлення початкових значень дозволяє значно пришвидшити навчання нейронної мережі, а в інших не має практично ніякого ефекту [20]. Проте негативного впливу від цієї дії нема, а необхідні обчислення є дуже простими та мають виконатись тільки на етапі побудови нейронної мережі, тому виконання ініціалізації не є зайвим.

2.2.4 Рівномірна ініціалізація

Основною задачею ініціалізації є відсутність зникаючих та вибухових градієнтів, тобто надмалих або надвеликих значень. Рівномірна ініціалізація

встановлює початкові значення таким чином, щоб при першому запуску всі вихідні значення нейронів були рівними одиниці. Значення кожного вагового коефіцієнта розраховується за наступною формулою:

$$w = \sqrt{\frac{1}{n_i}},$$

де n_i це кількість входів до кожного нейрона і-го шару.

Даний спосіб підходить для логістичної, випрямляючої лінійної чи експоненційної лінійної активаційних функцій.

Також при ініціалізації можна домножувати на випадковий коефіцієнт, що згенерований за рівномірним розподілом з середнім 0 для досягнення менш стабільного початкового розподілу. В результаті отримані ваги будуть краще динамічніше змінюватись на початкових етапах навчання.

2.2.5 Ініціалізація Ксав'єра

Ініціалізація Ксав'єра використовується для генерування початкових значень для гіперболічного тангенсу. Вони обчислюються за наступною формулою:

$$w = \sqrt{\frac{2}{n_i + n_{i+1}}}$$

де n_i це кількість входів до кожного нейрона i-го шару,

n_{i+1} це кількість входів до кожного нейрона $i + 1$ шару (кількість виходів i-го шару).

Аналогічно до попереднього способу ініціалізації, отримані значення можна домножувати на випадковий коефіцієнт, що згенерований за рівномірним розподілом з середнім 0 для досягнення менш стабільного початкового розподілу.

2.3 Висновки

У даному розділі розглянуто та описано алгоритми, що будуть використовуватись для класифікації текстів. Для баєсового класифікатора по суті використовується всього одна універсальна формула, в той час я в

нейронних мережах використовується комбінація матричних операцій та функцій активації.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розроблене програмне забезпечення складається з трьох основних компонентів:

- модуль для збору та підготовки навчальних та текстових даних. Основними його функціями є завантаження текстів з наданих джерел даних, їх розмічування та перетворення на формат, придатний для використання класифікатором. Також функції цього модулю використовуються при класифікації текстів, введених користувачем, коли перед їх безпосередньою класифікацію їх необхідно перетворити у відповідний формат;
- модуль класифікації, який містить реалізації алгоритмів класифікації, а також засоби для їх навчання та використання для класифікації даних, введених користувачем. У даному програмному комплексі присутні реалізації алгоритму наївного баєсова класифікатора, нейронних мереж прямого поширення та рекурентних нейронних мереж;
- модуль, що містить інтерфейс користувача. Включає в себе як засоби командного рядка, так і віконний інтерфейс для навчання класифікаторів та виконання класифікації заданих текстів.

В наступних пунктах розглянемо структуру та функції кожного модуля окремо.

3.1 Підготовка вхідних даних

Класифікація відноситься до машинного навчання з учителем, тому що для створення класифікатора необхідно мати набір попередню розмічених даних, на основі яких будуть знайдені закономірності для визначення класів. Таких даних необхідно багато, а тому створення корпусу навчальних текстів вручну є практично неможливою задачею. Через це необхідно згенерувати навчальні матеріали з доступних джерел, які вже надають інформацію про тексти та їх класи.

Для таких цілей чудово підходять онлайн-бібліотеки, які містять електронні версії наукових журналів. Оскільки такі тексти вже мають УДК, то

залишається лише завантажити опубліковані тексти, співвіднести їх з відповідними категоріями УДК та перетворити у формат, що буде використовуватись класифікатором для навчання та виконання класифікації.

3.1.1 Збір текстів

У якості джерела інформації було обрано національну бібліотеку України імені В. І. Вернадського. В її електронному архіві знаходиться близько семисот тисяч наукових публікацій.

На жаль, бібліотека не надає програмного інтерфейсу для отримання файлів, тому збір фалів було виконано за допомогою стандартного веб-інтерфейсу, що знаходиться за адресою "http://www.irbis-nbuv.gov.ua".

Першим етапом збору інформації є визначення списку статей для завантаження. Список опублікованих журналів можна отримати із алфавітного покажчика. Після цього для кожного журналу виконується перехід на відповідну сторінку, де відмічені всі номери, які є в наявності в архіві. Для кожного номеру виконується перехід на відповідну сторінку, яка містить посилання на опубліковані статті. Для кожної такої статті виконується завантаження документа в форматі pdf.

Після цього необхідно отриманий набір статей необхідно перетворити на текст та відсортувати відповідно до їх класів УДК. Для переведення pdf файлів до текстового формату була використана бібліотека PDFBox. Більшість документів, що були опубліковані в період з 2003 по 2011 рік містили відскановані сторінки замість тексту, тому видобути статті з них є досить складним завданням, адже методи розпізнавання тексту досі є недосконалими та забирають велику кількість часу та ресурсів. Оскільки отриманий корпус текстів був достатнього великий навіть без урахування таких документів, було вирішено виключити їх зі списку для створення навчальних даних.

Наступним етапом є отримання інформації про авторів та УДК. Проблемою отриманих статей є те, що вони представляють собою сторінки журналів, а тому початок документу не є початком статті, так само як і кінець документа не є кінцем статті. Тим не менш, через обмеження на мінімальний

розмір статей, що приймаються до публікації наявність двох статей на одній сторінці є неможливою, а тому необхідно лише знайти ознаки початку та кінця статті, і видалити всю інформацію, що міститься до початку та після кінця.

Початок статті визначається як співпадіння двох ознак:

- на першому рядку присутні три літери "УДК", а далі цифри та символи у відповідному форматі;
- на відстані не більше трьох рядків знаходиться рядок, що містить лише заголовні літери, такий рядок відповідає назві статті.

Кінець статті визначається як початок нової статті, відповідно до вищенаведеного правила, або кінець файлу, якщо наступна стаття не була знайдена.

Результатом роботи даного модуля є масив текстів, що відсортовані відповідно до категорій (класів УДК).

У програмній реалізації основними компонентами модуля є:

- структурні класи, що відповідають за збереження інформації про певні сутності: Category, Article, Edition, Journal;
- функціональні класи, що відповідають за розбір сторінок та отримання інформації з веб-сторінок, такі як VernadskyJournal, JournalDataExtractor, ArticleListExtractor, ArticleContentExtractor;
- крім того, тут міститься виконуваний файл ExportVernadskyLibrary, що зберігає всі необхідні налаштування та відповідає за запуск процесу експорту статей.

Діаграма класів даного модуля наведена на рисунку Рисунок 3.1.

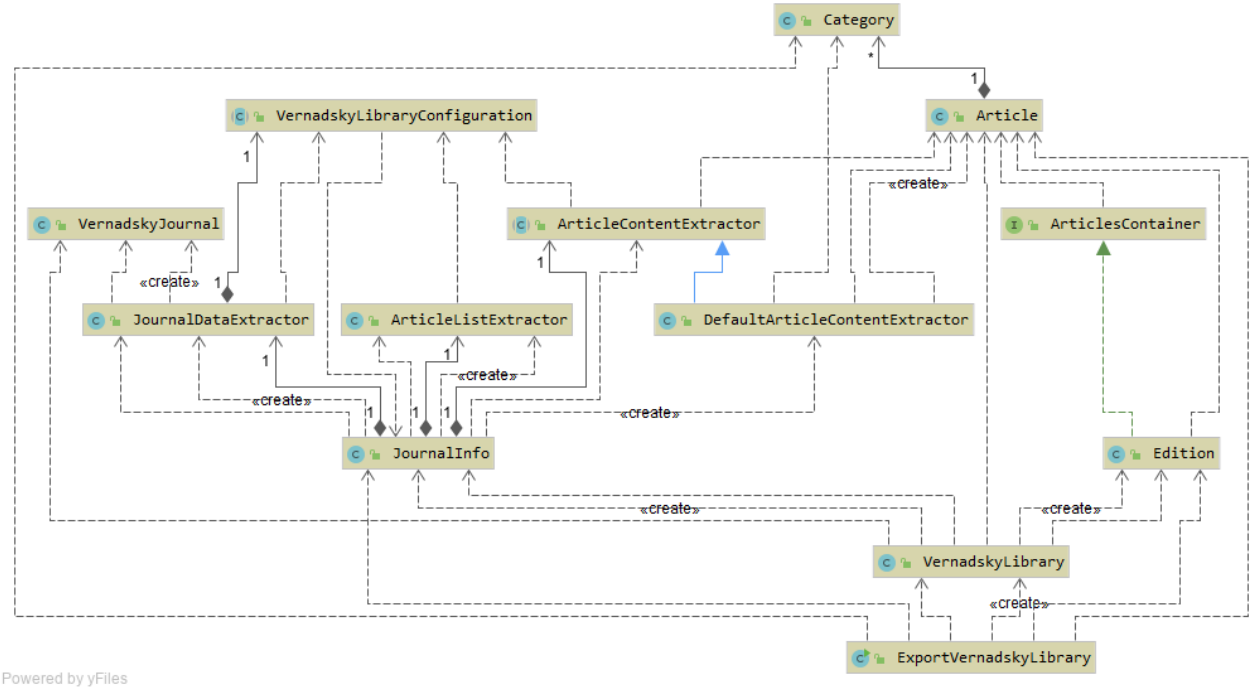

Рисунок 3.1 – Діаграма класів модуля збору текстів

3.1.2 Перетворення текстів

На другому етапі підготовки даних необхідно перетворити тексти у формат, придатний для програмного використання. Для спрощення роботи класифікатора необхідно прибрати з тексту незначущі символи, а також привести слова до базової форми, що дозволить значно зменшити варіативність вхідних даних та, відповідно, покращити точність класифікації.

На першому кроці видаляються всі літери, що не містяться в українському алфавіті, розділові знаки, а всі заголовні літери перетворюються на рядкові. Також на цьому етапі видаляється УДК, що був розпізнаний на попередньому кроці, щоб унеможливити використання класифікатором готової відповіді.

Наступним кроком перетворення є приведення слів до базової форми, адже такі дані як відмінок чи рід не несуть в собі даних відносно категорії та залежать виключно від лексичного контексту. Натомість, різні форми слова будуть розпізнаватись як різні семантичні одиниці, що ускладнить навчання класифікатора. Для пошуку єдиної форми слова використовується морфологічний аналізатор для української мови. Він визначає морфеми (структурні одиниці) слів та видаляє загальновживані частини, що не несуть

семантичного значення, наприклад, закінчення та деякі префікси, що містяться у спеціальному списку. Після цього з тексту видаляється всі слова довжиною в одну та дві літери, тому що в українській мові такими словами в основному є прийменники та займенники, які також не несуть семантичної інформації.

Після виконання відповідних перетворень були видалені всі документи, що в результаті перетворень містили менше двохсот слів (що приблизно відповідає половині сторінки друкованого тексту 14 кеглем). В цю категорію попали тексти, що написані іноземними мовами, або мали проблеми з кодуванням.

Діаграма класів для даного компоненту зображена на рисунку Рисунок 3.2.

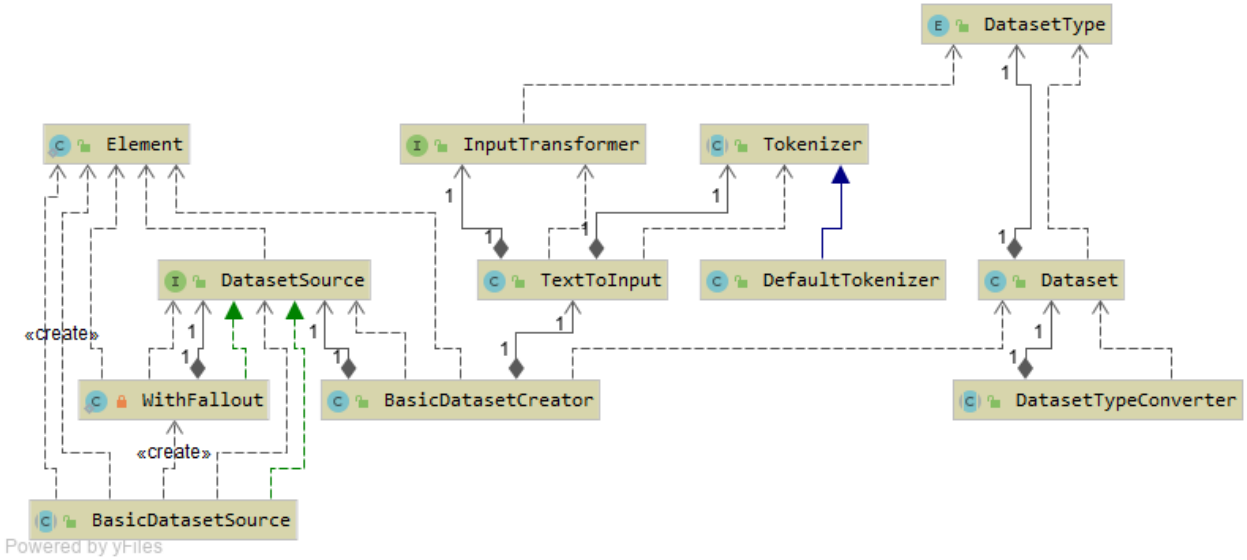

Рисунок 3.2 – Діаграма класів пакету створення датасету

В результаті даних перетворень було отримано 159546 статей, розподілених між 842 категоріями. Цей корпус текстів став базою для подальшого генерування навчальних та тестових прикладів для оцінки роботи різних алгоритмів класифікації.

3.1.2.1 Модель Bag of Words

Bag of Words це найпростіша модель представлення текстів. В для кожного слова, що входить в текст зберігається кількість таких входжень. Спершу необхідно сконвертувати слова до числового представлення замість текстового. Для цього необхідно побудувати словник, в якому зібрати всі слова, що зустрічаються в текстах. Після цього пронумерувати слова по порядку, в

результаті чого кожне слово, що зустрічається в текстах буде мати унікальний номер. Після цього, для кожного тексту обчислюється представлення тексту, яке буде використовуватись при класифікації. Це масив, що на *i*-й позиції містить кількість входжень *i*-го слова в тексті.

Перевагою такого представлення є простота. Крім того, він значно зменшує розмір вхідних даних, що покращує ефективність деяких класифікаторів, наприклад, наївного баєса, а крім того, значно скорочує обчислювальні витрати на їх навчання. Основними його недоліками є втрата контексту між словами, адже у результуючому представленні вони зберігаються неупорядковано, та неможливість обробляти слова, які не зустрічалися при початковій побудові словника.

3.1.2.2 Модель TF IDF

Покращенням моделі Bag of Words є TF IDF (від англ. TF — term frequency, IDF — inverse document frequency). Простий підрахунок кількості слів, як це зроблено у Bag of Words, приводить до поганих результатів для текстів різної довжини. Виявляється, що з точки зору множини слів, менший текст є підмножиною більшого тексту, хоча й вони належать до різних категорій. Це стається через те, що статистично в більшому тексті завжди більше різноманітних слів, навіть якщо вони зустрічаються нечасто. Для уникнення цієї проблеми використовується модель TF IDF. Аналогічно до Bag of Words, тут використовуються множина слів, до кожного з яких співставляється число, що позначає важливість цього слова в тексті. Проте замість підрахунку кількості входжень для кожного слова використовується частка двох метрик – частоти цього слова в заданому тексті та кількості документів, що містять це слово. В результаті метрика стає більшою при збільшенні входження слова до даного документу та меншою при збільшенні документів, до яких це слово входить. Це дозволяє зменшити вагу слів, що зустрічаються в різних текстах, та збільшити вагу слів, що часто зустрічаються в небагатьох текстах. Інші переваги та недоліки зберігаються з моделі Bag of Words – нове представлення також

залишаються відносно простим, проте можливість обробляти контекст слів в ньому відсутня.

Через те, що необхідні числові значення вже були підраховані на етапі генерування моделі Bag of Words, то для створення датасету даного типу немає необхідності генерувати всі дані з початку. Якщо попередньо для кожного слова обчислити кількість документів, в яких воно зустрічається, то документи можна генерувати одразу з вхідних даних моделі Bag of Words. Частота слова обчислюється як кількість входжень даного слова, поділена на загальну суму токенів; частота документів була обчислена на першому етапі. Частка цих двох значень і буде шуканим числом.

За рахунок використання Bag of Words у якості бази для генерування даного датасету зменшується фізичний розмір файлів для зберігання на диску, адже замість чисел з плаваючою комою, що потребують вісім байт, можна зберігати лише число входжень слова в текст, для чого достатньо одного байта, що призводить до зменшення розміру датасету в вісім разів.

3.1.2.3 Масив токенів

Масив токенів представляє собою числові значення для слів в тексті без подальшої обробки. В результаті з’являється можливість опиратись на контекст слова при класифікації, проте довжина такого масива рівна кількості слів у початковому тексті, тому для класифікатора, який буде обробляти ці дані, з’являється додаткова вимога – обробляти вхідні дані змінної довжини, адже всі тексти різні за розміром. Наразі лише рекурентні нейронні мережі мають таку можливість.

3.1.3 Збереження отриманих даних

Для пришвидшення роботи з текстами, як при генеруванні датасетів, так і при навчання класифікаторів доцільно зробити дві оптимізації. По-перше, файли містять довгі послідовності нулів, адже кожен з текстів містить лише невелику частку загального словникового запасу. По-друге, доцільним є зберігання всіх текстів у одному файлі, замість використання окремого файлу для кожного тексту. Для досягнення цих задач навчальні та тестові дані зберігається у форматі

`.tar.gz`, тобто заархівовані алгоритмом tape archive та стиснуті за допомогою алгоритму deflate. В результаті розмір текстів зменшується майже в тридцять разів (з 1456,3 до 51,2 мегабайтів), а швидкість обробки файлів, наприклад, при конвертації з формату Bag of Words до TF IDF, зменшується з двадцяти до чотирьох хвилин.

3.2 Реалізація алгоритмів класифікації

Зовнішній інтерфейс готового класифікатора має приймати текст та у відповідь видавати список категорій та ймовірність належності тексту до кожної з них. Проте самі класифікатори приймають на вхід масив токенів, та видає масив ймовірностей. Відповідно для надання зручного інтерфейсу користувачам був створений клас GeneralClassifier, який приймає текст, токенізує його, відправляє на класифікатор та перетворює отриману відповідь на формат для зовнішнього користувача.

3.2.1 Наївний баєсів класифікатор

3.2.1.1 Рушій Apache Spark.

Для роботи з баєсовим класифікатором було використано рушій Apache Spark. Spark це рушій для обробки великих даних з відкритим вихідним кодом та вільною ліцензією, розроблений організацією Apache Software Foundation. Для оброблення даних на основі Spark можуть використовуватись такі мови, як Java, Scala, Python чи R. За допомогою них можна створювати задачі, які будуть виконуватись на розподіленому кластері.

Наразі Apache Spark є найпродуктивнішою обчислювальною системою загального призначення. Наприклад, за рахунок кешування він працює від 10 до 100 разів швидше порівняно із використанням MapReduce на основі Hadoop без додаткових модифікацій. Він широко використовується дослідниками з усього світу та в провідних світових компаніях, таких як Alibaba, Cloudera, Databricks, IBM, Intel, Yahoo, Cisco Systems тощо. Крім того, він підтримує роботу з розрідженими векторами в якості вхідних даних, що покращує продуктивність

роботи класифікатора, адже як було показано вище, вхідні масиви містять значну кількість нульових значень.

Основним його компонентом є драйвер, який займається зберіганням основної інформації та стану програми, обробкою та виконанням запитів користувача та аналізом і розподілом обчислювальних задач. Крім нього, є певна кількість виконавців, які займаються виконанням обчислень, отриманих від драйвера. За рахунок використання мови Java для створення програми, рушій Spark можна розгорнути в тій же віртуальній машині, що й інші компоненти програми, що значно спрощує розробку та пришвидшує роботу класифікатора.

3.2.1.2 Навчання та використання класифікатора

Для навчання класифікатора використовуються розріджені вектори з метою економії пам'яті. Для їх створення використовується стандартний датасет, дані з якого конвертуються у потрібний формат. Також проблемою для класифікаторів є проблема перенавчання, тому при навчанні можна вказати параметр, який визначає, яку частку вхідних даних необхідно використовувати для навчання. Діаграма класів, що використовуються при роботі з баєсовим класифікатором зображена на рисунку Рисунок 3.3.

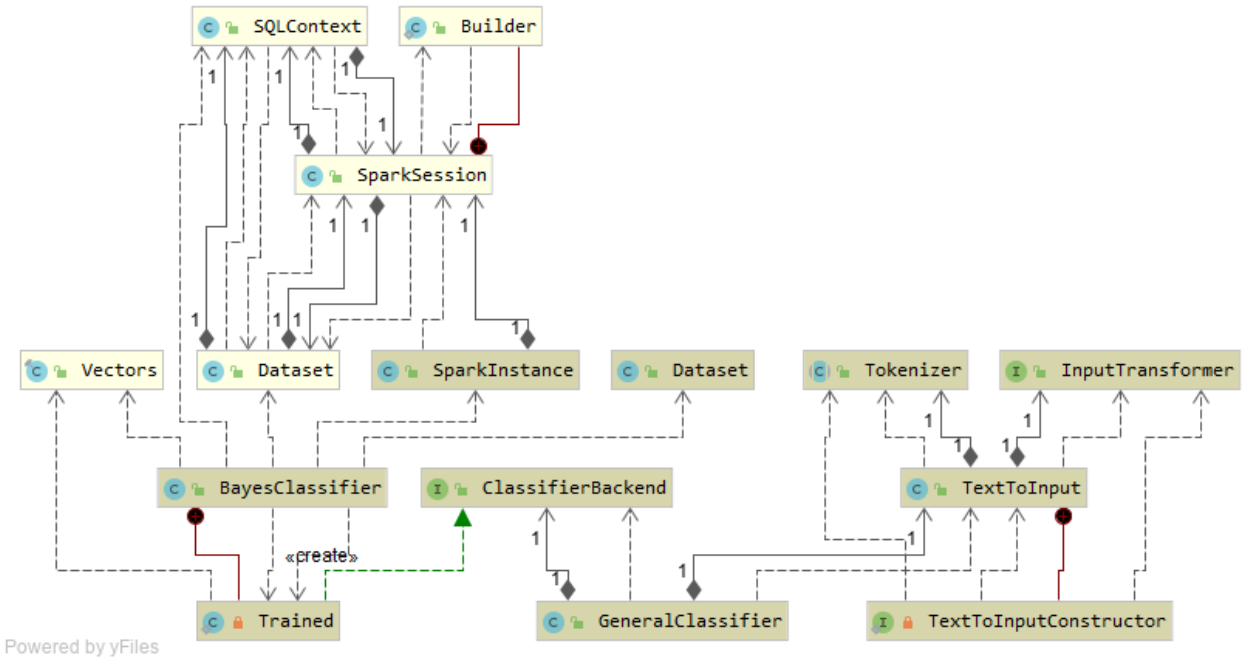

Рисунок 3.3 – Діаграма класів для баєсового класифікатора

В результаті для використання баєсового класифікатора достатньо створити екземпляр класу BayesClassifier та передати датасет у якості параметра в метод train. Після цього класифікатор буде готовий до роботи.

3.2.2 Нейронні мережі

3.2.2.1 Програмне забезпечення для навчання нейронних мереж

Нейронні мережі були винайдені та теоретично обґрунтовані наприкінці минулого століття, проте розвиток отримали лише недавно, через зростання обчислювальних потужностей. Навіть зараз, вони потребують спеціального обладнання та програмного забезпечення для своєї роботи. Для роботи з нейронними мережами на мові Java була використана бібліотека deeplearning4j. Ця бібліотека включає в себе реалізації алгоритмів навчання нейронних мережам прямого поширення та рекурентними нейронним мережами. За рахунок використання Apache Spark для розподілення та виконання задач при навчанні мереж досягається продуктивність, порівняна з бібліотекою Caffe, яка є де-факто стандартом для навчання нейронних мереж. Проте за рахунок можливості виконувати розподілення обчислень засобами Apache Spark потенційна швидкість роботи є значно вищою, за умови наявності необхідних обчислювальних ресурсів.

Для використання апаратного прискорення при роботі з нейронними мережами застосовується програмне забезпечення Nvidia CUDA. Це додаткові драйвери, які надають низькорівневий інтерфейс для виконання обчислень на графічних процесорах. Через використання великої кількості ядер у графічних процесорах, такі обчислення виконуються значно швидше, ніж на центральному процесорі. deeplearning4j в комбінації з драйверами CUDA дозволяє виконувати практично будь-які операції, що можна представити у вигляді графа обчислень, в тому числі операції з навчання нейронних мереж.

3.2.2.2 Структура та параметри нейронних мереж

Побудова структури нейронних мереж виконується в класі FeedForwardNeuralNetwork. Він створює конфігурацію нейронної мережі

прямого поширення відповідно до отриманої конфігурації. Значеннями, що можна задати при створенні нейронної мережі є:

- layers – кількість прихованих шарів нейронної мережі. В загальному випадку збільшення кількості шарів приводить до збільшення точності класифікації, проте така мережа потребуватиме більше тренувальних прикладів та часу для навчання;
- neuronActivation – функція активацій нейронів для проміжних шарів. Доступними функціями є: логістична функція, гіперболічний тангенс, випрямлена лінійна функція та експоненційна лінійна функція;
- lossFunction – функція для обчислення втрат. Використовується при обчисленні точності нейронної мережі в процесі навчання.

Наступні значення також можна задати при створенні нейронної мережі, проте вони впливають не на мережу, а на процес навчання. Вони є оптимальними для обраного типу мереж і змінювати їх без необхідності не рекомендується;

- outputActivation – функція активації останнього шару нейронів. Оскільки ця нейронна мережа використовується для мультикласової класифікації, то на виході очікується масив ймовірностей, які в сумі будуть давати 100%. За замовчанням тут використовується нормована експоненційна функція, яка і виконує відповідне перетворення;
- seed – значення ініціалізації для генератора випадкових чисел, що використовується для генерації початкових значень нейронної мережі, розподілу текстів між навчальними та тестовими прикладами та перемішування навчальних даних. Через використання випадковостей навчання та результати нейронної мережі можуть відрізнятись, тому для отримання послідовних результатів необхідно використовувати одне й теж значення;
- learningRate – швидкість навчання нейронної мережі. Збільшення цього параметру призводить до пришвидшення навчання, проте великі значення цього параметру можуть призвести до неможливості знаходження локального мінімуму при виконанні градієнтного спуску, що призведе до гірших результатів;

- epochs – максимальна кількість навчальних епох, кожна з яких є повним проходом по доступним тестовим даним. Використовується для припинення навчання, якщо воно триває надто довго. Зазвичай процес закінчується раніше через перенавчання, тому якщо цей максимум епох був досягнутий в процесі навчання є проблеми – або в коді програми присутній дефект, що призводить до нескінченного навчання, або швидкість навчання нейронної мережі є надто малою, тому треба або збільшити швидкість навчання, або збільшити кількість епох;
- maxNonImprovementEpochs – кількість епох без покращень, після яких навчання припиняється. Зазвичай погіршення точності після чергової епохи сигналізує про початок перенавчання, що призводить до погіршення загальної точності класифікації. Проте в деяких випадках це може призвести до знаходження нового локального максимуму. Збільшення цього параметру може привести до знаходження нових коефіцієнтів, які будуть ефективнішими за попередній максимум, проте загалом така ситуація зустрічається набагато рідше, ніж перенавчання, а збільшення кількості епох приводить до збільшення часу навчання до отримання фінальної версії мережі;
- testsPart – частка даних, що будуть використовуватись у якості тестових. Збільшення цього параметру приводить до спрощення оцінки якості нейронної мережі, проте через недостатню кількість навчальних прикладів нейронна мережі може дуже швидко перенавчатись;
- batchSize – кількість навчальних текстів, після яких буде виконано перерахунок коефіцієнтів нейронної мережі. Збільшення цього параметру збільшує точність навчання нейронної мережі, проте зменшує його швидкість.

Якщо деякі параметри не були передані використовується значення за замовчанням. Список таких значень наведений у таблиці 3.1.

Таблиця 3.1 – Значення за замовчанням для нейронної мережі прямого поширення

Параметр	***Значення***
layers	1
neuronActivation	SIGMOID (сигмоїдальна функція)
lossFunction	NEGATIVELOGLIKELIHOOD (негативна функція логарифмічної правдоподібності)
outputActivation	SOFTMAX (нормована експоненційна функція)
seed	387872891
learningRate	0.1
epochs	100
maxNonImprovementEpochs	5
testsPart	0.3
batchSize	32

Значення вихідного шару нейронної (функція SOFTMAX) обчислюється за наступною формулою:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{n=1}^{N} e^{z_n}}, i = 1..N.$$

В результаті якими б не були значення, обчислені нейронною мережею, на виході завжди будуть ймовірності. При цьому більші значення перетворяться на більші ймовірності, тому при виборі найбільш ймовірного класу значення буде однаковим як до перетворення, так і після.

Кількість нейронів на кожному шарі обчислюється за наступною формулою:

$$count_i = (\frac{input}{output})^{1-\frac{i}{n+1}} * output$$

Де

$count_i$ – кількість нейронів у i-му шарі,

$input$ – розмір вхідного вектора,

$output$ – розмір вихідного вектора.

Процес навчання нейронної мережі виконується наступним чином:

а) для кожного навчального прикладу з поточного тренувального пакету

1) вектор значень що репрезентує навчальний приклад подається на вхід до нейронної мережі;

2) визначити помилку як $d = (target - actual) * f'(input)$;

3) обчислити необхідну зміну ваги нейрону як $w = w + learning\_rate * d$;

4) передати d до наступного шару нейронної мережі на виконати аналогічні перетворення там

б) після завершення проходу по тренувальному пакету обчислити змінукоефіцієнтів нейронної мережі як середнє всіх значень, що були обчислені на кроці 3 та обчислити нові коефіцієнти для кожного шару.

Оскільки при такому підході коефіцієнти оновлюються не для кожного навчального прикладу, а лише раз на один пакет (за замовчанням – 32 приклади), то першу частину алгоритму з обчислення реальної відповіді мережі та обчислення необхідної зміни коефіцієнтів можна розпаралелювати, за рахунок чого досягається підвищення швидкості навчання. Крім того, при такому підході згладжується негативний вплив некоректних тренувальних прикладів, які можуть зустрітись у оброблюваних даних.

3.3 Створення інтерфейсу користувача

Для навчання класифікаторів було створено консольний інтерфейс, тому що виконання цієї роботи спроектовано для запуску на обчислювальних кластерах, які зазвичай мають лише консольний інтерфейс. Проте для виконання власне класифікації необхідно надати два інтерфейси: по-перша, користувач повинен мати можливість виконувати класифікацію зі зручного графічного інтерфейсу; по-друге, повинен бути зручний програмний інтерфейс для автоматичного виконання класифікації, наприклад у випадку коли такий сервіс буде використовуватись бібліотеками для класифікації електронних надходжень.

Одним з найбільш розповсюджених програних інтерфейсів наразі є REST. При використанні JSON для передачі даних ми отримуємо сучасний програмний інтерфейс, що підтримується більшістю мов програмування. Відповідно, для створення графічного інтерфейсу найбільш доцільним буде створення веб-сайту, тому що він зможе використовувати той же інтерфейс для виконання класифікації та потребуватиме лише розробки веб-сторінки, на який можна ввести текст та отримати відповідь. Іншою перевагою веб-сайту порівняно з окремою програмою є відсутність вимог до комп'ютера користувача – якщо в нього працює веб-браузер то користувач зможе виконувати класифікацію без необхідності, наприклад, встановлювати спеціалізовані драйвери для CUDA обчислень, що необхідні для виконання обчислень для нейронних мереж.

3.3.1 Програмний інтерфейс

У якості веб-серверу був обраний сервер Netty. На відміну від інших серверів для Java, таких як Apache Tomcat, Oracle Weblogic або WildFly, Netty містить лише код для роботи з HTTP з'єднаннями без додаткових функцій для обробки транзакцій, впровадження залежностей, роботи з чергами повідомлень та іншими функціями, що передбачені специфікацією JEE. Оскільки в даній програмі для збереження інформації про класифікатори використовується файлова система, а додаткових інтерфейсів взаємодії з користувачем та іншими програмами не передбачено, то наявність будь-яких інших функцій крім обробки REST запитів призведуть лише до сповільнення роботи програми та не призведуть до жодних покращень.

На початку роботи сервер з'єднується з сервісом Apache Spark, який буде виконувати обчислення, необхідні для класифікації, а якщо сервіс не знайдено, то запускає його самостійно. На наступному кроці у пам'ять завантажуються натреновані класифікатори, що були вказані при запуску сервера, після чого сервер є готовим до обробки клієнтських запитів.

Даний сервер надає всього два методи:

- отримання списку доступних класифікаторів;
- класифікація введеного тексту.

Для отримання списку класифікаторів достатньо виконати GET запит на адресу */api/classification/classifiers*. В результуючому JSON повідомлені знаходиться список доступних класифікаторів. Приклад такого запиту зображений на рисунку Рисунок 3.4 – Отримання списку класифікаторів.

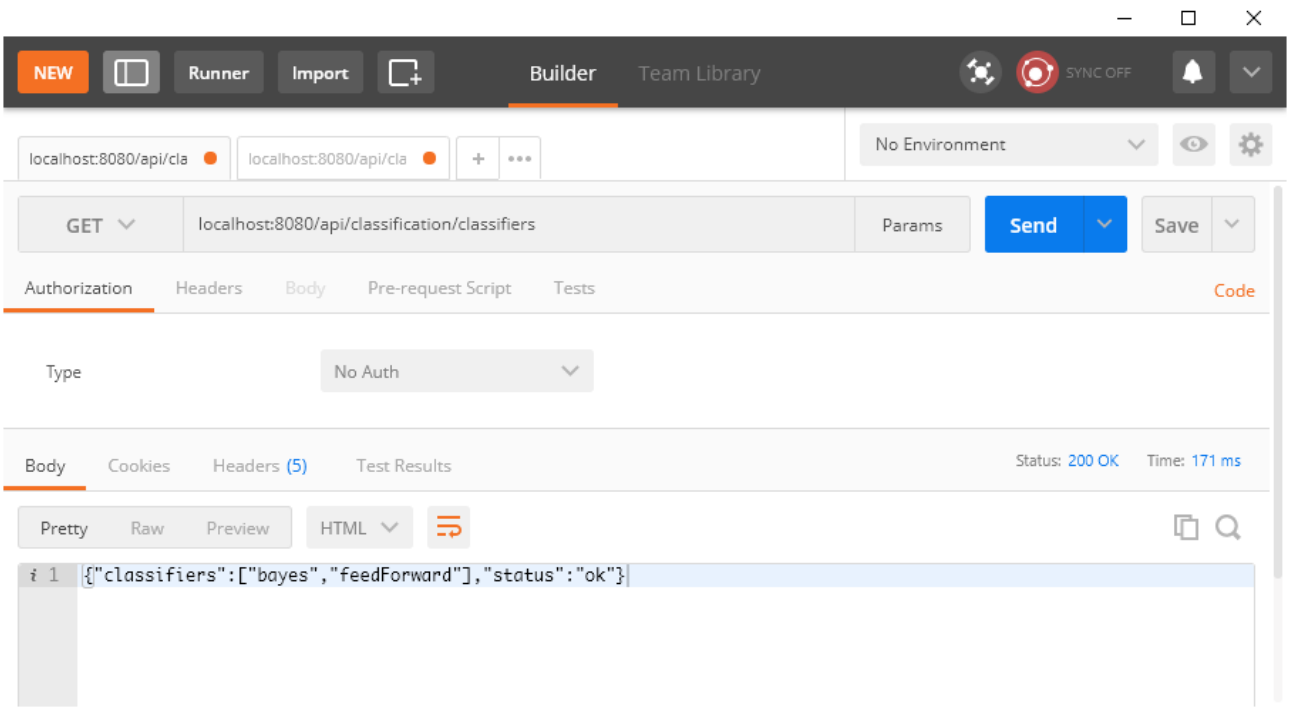

Рисунок 3.4 – Отримання списку класифікаторів

Для виконання класифікації необхідно надіслати POST запит на адресу */api/classification/classify*. У запиті необхідно вказати ім’я класифікатора та текст, який необхідно класифікувати. У відповіді від сервера знаходиться список категорій, для кожної з яких зазначена ймовірність належності тексту до неї. Також у відповіді від сервера наявний номер УДК, до якого відноситься заданий текст, якщо цей номер вдалося розпізнати в тексті. Приклад такого запиту зображено на рисункуРисунок 3.5.

Для виконання класифікації необхіно спочатку виконати очистку тексту та перетворення його на вектор вхідних даних для класифікатора. Ці операції виконуються за допмогою класів Tokenizer та TextToInput, що використовувались при підготовці навчальних даних.

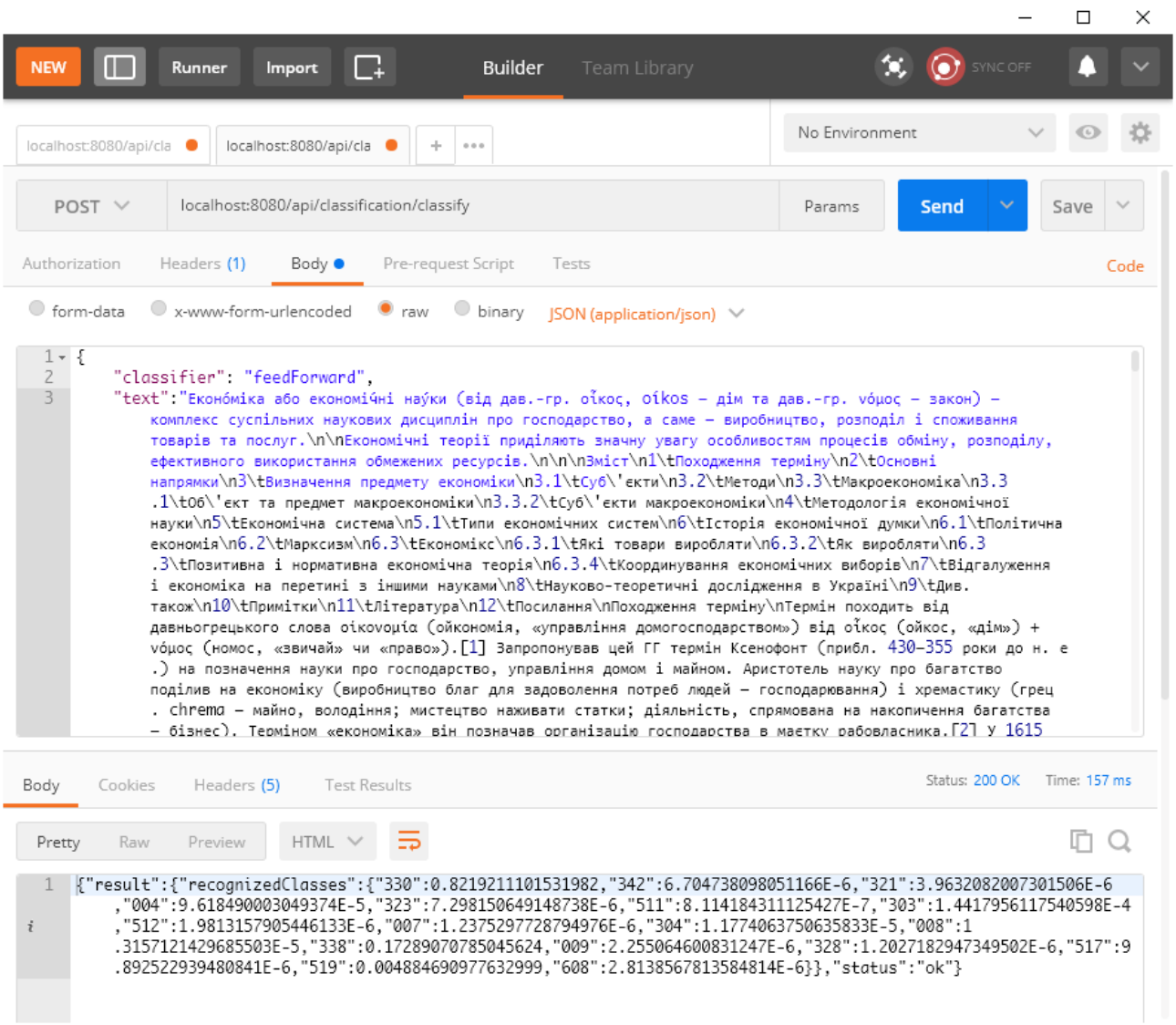

Рисунок 3.5 – Класифікація тексту

Загальна структура класів що використовуються для створення програмного інтерфейсу зображена на рисунку Рисунок 3.6.

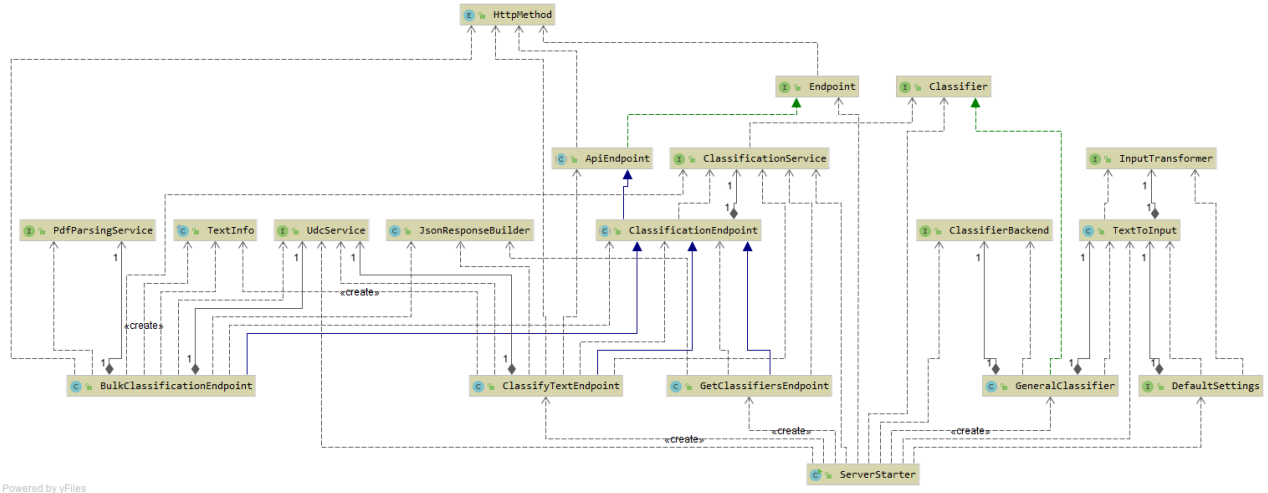

Рисунок 3.6 – Діаграма класів, що використовуються в веб-сервері

3.3.2 Веб-інтерфейс

Веб-інтерфейс складається з однієї сторінки, на який можна обрати класифікатор та виконати класифікацію тексту.

Проте сама сторінка є лише шаблоном для заповнення інформацією. Для заповнення цих шаблонів використовується фреймворк Vue.js. Він дозволяє створювати користувацькі інтерфейси на основі шаблону Model-View-ViewModel (MVVM). Ця концепція дозволяє розділити дані та їх представлення таким чином, щоб їх можна було модифікувати без негативного випливу на інший компонент. Разом з тим, вона дозволяє виконувати так зване «зв’язування» даних, тобто синхронізацію даних та інтерфейсу. Це одночасно дає дві можливості – по-перше, вигляд інтерфейсу на відображену на ньому інформацію можна змінювати просто змінивши значення в моделі даних. По-друге, можна синхронізовувати поля введення на інтерфейсі з відповідними змінними моделі даних, тобто інформація, що була введена користувачем, стає одразу доступною для обробки програмою. Цей фреймворк заснований на реактивній поведінці, тобто оновлення інтерфейсу та даних відбувається при виникненні певних подій, наприклад, при надходженні нової інформації з серверу або виникненні нового введення від користувача (заповнення поля, натискання кнопки тощо). Таким чином даний підхід дозволяє значно спростити розробку інтерфейсу для даного програмного застосунку, тому що зникає

необхідність ручної модифікації HTML коду, адже при завантаження нових даних вони автоматично будуть підставлені у відповідні поля графічного інтерфейсу.

Для відправки повідомлень та отримання результату від серверу використовується плагін jQuery. Він дозволяє зручним чином надсилати дані на сервер, отримувати відповідь та обробляти помилки. Також її можна використовувати для прямої маніпуляції об’єктами, що знаходяться на сторінці, але використання такого підходу не рекомендується, тому що це призводить до змішування двох підходів до роботи з інтерфейсом, тому що в проекті вже використовується бібліотека Vue.js.

Від сервера на веб-сторінку надсилаються лише ймовірності належності для кожної категорії, а фінальна класифікація виконується вже на стороні клієнта.

Приклад виконання класифікації з використанням графічного інтерфейсу наведено на рисунках Рисунок 3.7 – Рисунок 3.8.

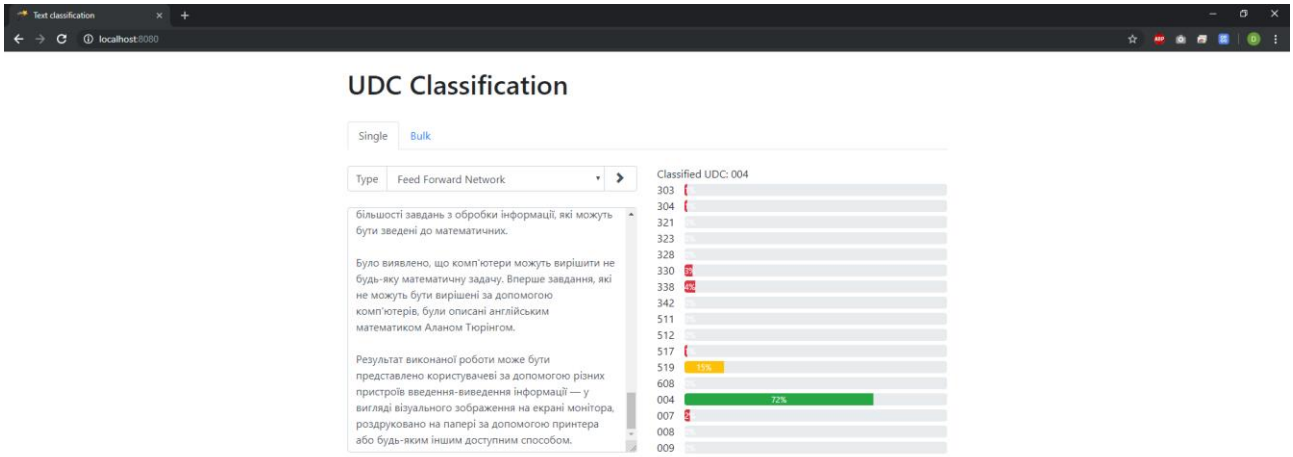

Рисунок 3.7 – Класифікація тексту комп’ютерної тематики

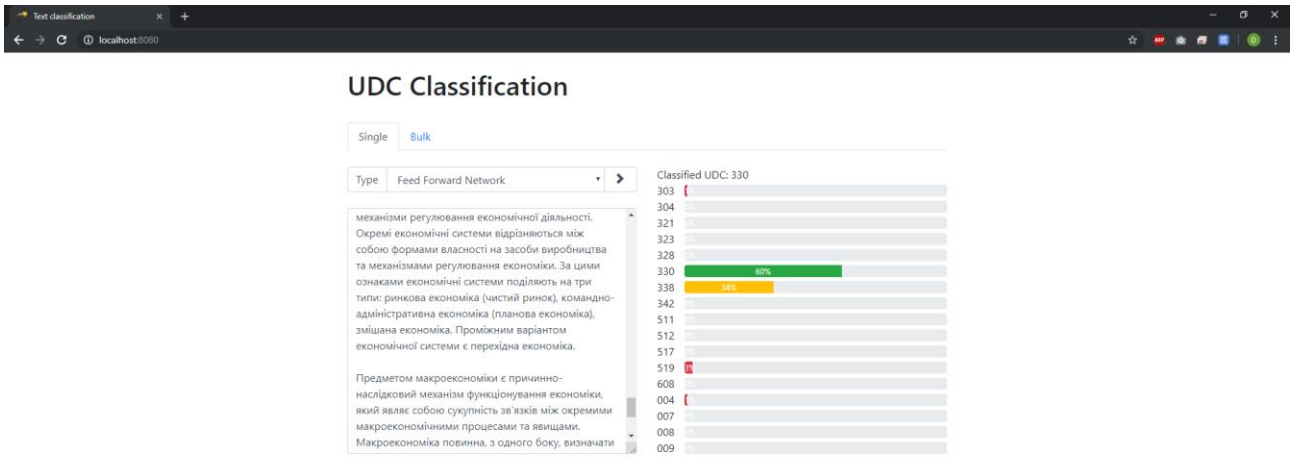

Рисунок 3.8 – Класифікація тексту, що частково належить до суміжної категорії

Належність до кожної з категорій визначається наступним чином: для всіх категорій обирається максимальна ймовірність. Вважається, що текст належить до категорії, якщо ймовірність належності відрізняється від максимуму не більше, ніж на тридцять відсотків, приклад такої класифікації зображено на рисунку Рисунок 3.9.

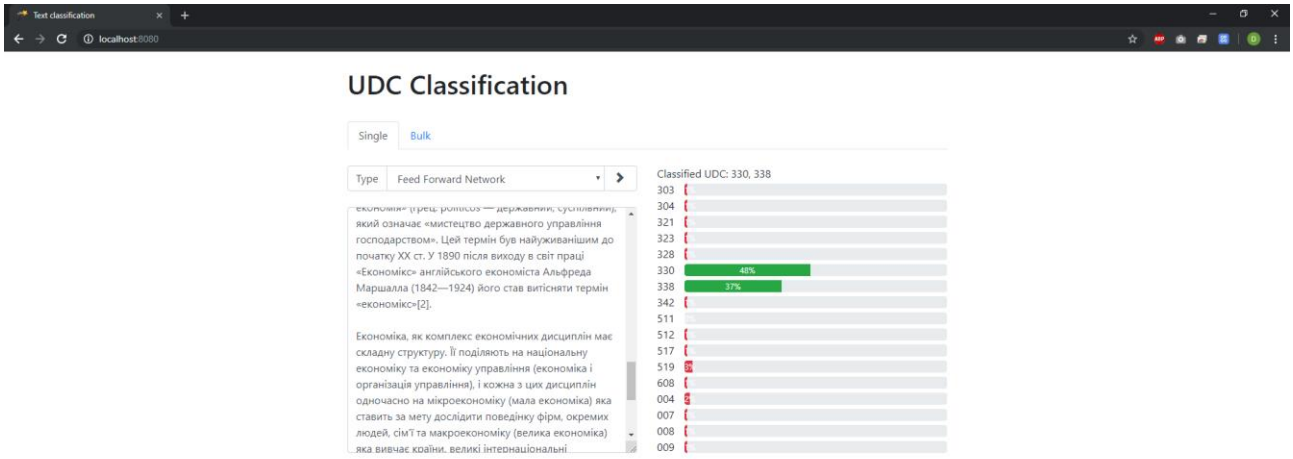

Рисунок 3.9 – Класифікація тексту, що належить до двох категорій

Категорії, до яких належить текст, позначаються зеленим кольором для полегшення візуального сприйняття. Жовтим кольором позначаються категорії, ймовірність належності до яких більше десяти відсотків. Інші категорії позначаються червоним кольором. Якщо ймовірність для декількох класів відрізняється не більше ніж на тридцять відсотків, то всі вони позначаються зеленим кольором.

3.4 Висновки

Розроблене програмне забезпечення складається з трьох основних компонентів, які відповідають за збір та нормалізацію навчальних даних, навчання класифікаторів та виконання класифікації текстів, введених користувачем.

Для збору даних для навчання класифікаторів був використаний електронний архів Національної бібліотеки України імені В. І. Вернадського. З завантажених документів були видобуті статті та їх класи, після чого отримані дані було використано для навчання наївного баєсового класифікатора на нейронних мереж. Для виконання обчислень була використана система Apache Spark, яка дозволяє розподілення навчальних задач у декілька потоків та драйвери CUDA, які дозволяють перенести частину обчислень на графічний процесор, значно прискорюючи швидкість навчання.

Для використання класифікаторів доступний програмний інтерфейс, доступний за протоколом HTTP. Він надає операції для отримання списку доступних класифікаторів та класифікації введеного тексту. На основі цього інтерфейсу була розроблена веб-сторінка, яка за допомогою створеного API надає зручний графічний інтерфейс для класифікації текстів.

4 РЕЗУЛЬТАТИ КЛАСИФІКАЦІЇ

Для навчання та тестування класифікаторів було використано тексти, зібрані з бібліотеки Вернадського. У якості категорій для класифікації було обрано двадцять найбільш популярних категорій УДК:

- 004 – Комп'ютерна наука та технологія. Застосування комп'ютера;
- 316 – Соціологія;
- 330 – Економіка в цілому;
- 331 – Праця. Працевлаштування. Робота. Економіка праці. Організація праці;
- 332 – Регіональна економіка. Територіальна економіка. Економіка землі. Економіка житла;
- 336 – Фінанси;
- 338 – Економічне становище. Економічна політика. Управління та планування в економіці. Виробництво. Послуги. Ціни;
- 339 – Торгівля. Міжнародні економічні відносини. Світова економіка;
- 351 – Напрями діяльності органів державного адміністративного управління;
- 378 – Вища освіта. Університети. Підготовка наукових кадрів;
- 519 – Ймовірність. Математична статистика. Обчислювальна математика. Математична кібернетика;
- 612 – Фізіологія. Фізіологія людини та порівняльна фізіологія;
- 615 – Фармакологія. Терапія. Токсикологія;
- 616 – Патологія. Клінічна медицина;
- 621 – Загальне машинобудування. Ядерна техніка. Електротехніка. Машинобудування в цілому;
- 631 – Загальні питання сільського господарства;
- 633 – Рільництво. Польові сільськогосподарські культури та їх виробництво;
- 636 – Загальні питання тваринництва. Розведення тварин і птахів. Скотарство. Домашні тварини та їх розведення;

- 657 – Бухгалтерія. Бухгалтерський облік. Рахівництво;
- 658 – Організація виробництва, менеджмент. Економіка підприємств. Організація та техніка торгівлі.

Всього було зібрано 70170 статей, з яких 70% було використано для навчання класифікаторів та 30% для їх тестування.

4.1 Наївний баєсів класифікатор

Навчання баєсова класифікатора виконувалось на процесорі Intel i7-9700k. Проте вузьким місцем, що сповільнювало навчання став жорсткий диск, який був завантажений на 100% при завантаженні процесора в 35%. Загальний час навчання – близько двадцяти хвилин.

Отриманий класифікатор правильно визначає категорію лише для 45,70% текстів. При цьому зменшення кількісті навчальних прикладів також зменшує точність класифікації, що показує відсутність перенавчання.

4.2 Нейронні мережі

При навчанні нейронних мереж було досліджена ефективність класифікації при зміні кількості прихованих шарів та функції активації нейронів.

Для дослідження впливу кількості прихованих шарів на ефективність класифікації інші параметри були зафіксовані з наступними значеннями: у якості функції активації було використано сигмоїдальну функцію, а мультиплікатор швидкості навчання був рівний 0,1. Отримані результати наведено в таблиці 4.1.

Таблиця 4. 1 - Дослідження впливу кількості прихованих шарів

Кількість шарів	*Час навчання нейронної мережі, хв.*	*Точність класифікації, %*
1	228	43,03%
2	287	47,66%
3	355	48,10%
4	420	47,94%
5	483	48,12%

Отримані результати показують, що після двох шарів ріст точності класифікації є досить незначним, але час, необхідний для навчання мережі лінійно зростає.

Для проведення подальшого тестування було використано мережу з двома прихованими шарами. При зміні функцій активації результати є наступними:

Таблиця 4.2 - Дослідження впливу функції активації

Функція активації	*Точність класифікації, %*
Сигмоїдальна	47,66%
Гіперболічний тангенс	47,60%
Випрямлена лінійна	47,57%

Результати свідчать, що у даному випадку хоча сигмоїдальна функція і показує кращі результати, проте в цілому вибір функції активації не сильно впливає на остаточний результат.

4.3 Ієрархічний класифікатор

Під час аналізу помилок класифікації було виявлено, що більшість неправильно класифікованих текстів було віднесено до суміжної категорії. Наприклад тексту з категорії 336 «Фінанси» присвоюється категорія 339 «Торгівля». Найбільше таких помилок присутні у групах, що умовно можна назвати «Економіка» та «Математика і алгоритми». Для виконання класифікації для цих категорій було навчено дві окремі нейронні мережі, що досягли точності класифікації в 73,22% для економічних категорій та 81,65% для математичних. Для порівняння, при використанні однієї мережі точність класифікації становила 45,44% та 46,54% відповідно. Підхід з використанням ієрархії класифікаторів дозволив підвищити загальну точність класифікації з 47,66% до 59,74%.

Загальні підсумки ефективності класифікаторів зображені на рисунку 4.1.

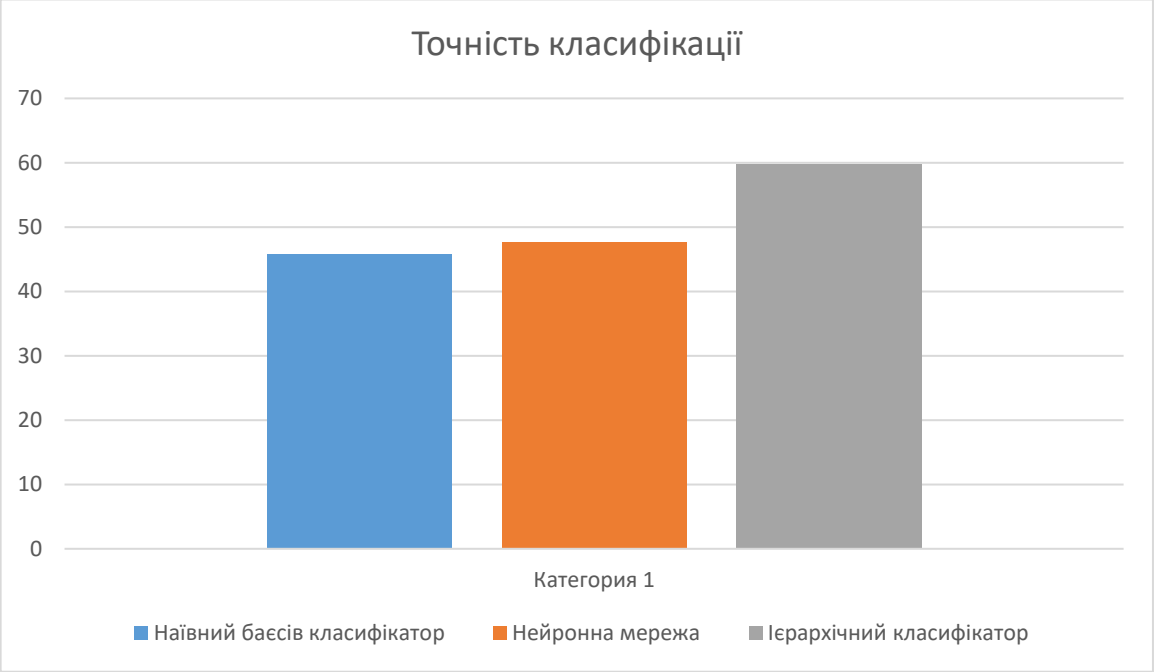

Рисунок 4.1 – Порівняння класифікаторів

4.4 Мультикласова класифікація

Для виконання мультикласової класифікації замість простого обрання максимальної ймовірності необхідно визначити поріг ймовірності для визначення класів. Таким чином, всі класи, ймовірність належності до яких буде вища за задану відповідають даному тексту. Для визначення цього порогу було обрано 5816 текстів, що належать мінімум до двох категорій. Ймовірність було підібрано для мінімізації суми кількості категорій, що було помилково пропущені та помилково призначені текстам. В результаті порогове значення ймовірності прийнято в 9,5%.

4.5 Визначення некоректної ручної класифікації

При ручному аналізі некоректно розпізнаних текстів було виявлено, що категорія, зазначена автором зазвичай знаходиться на другому чи третьому місці. При цьому категорії, що були визначені класифікатором також є коректними. Наприклад для статей, що описують застосування математичних методів в економіці класифікатор в першу чергу призначає категорію 004 або 519, що відповідають власне математичним методам та алгоритмам. Автори, в більшості випадків, відносять таку статтю лише до економіки, через що оцінка точності

класифікатора є значно нижчою ніж реальна якість класифікації. Якщо вважати правильно класифікованим текст, що містить категорію, зазначену автором, серед трьох найбільш ймовірних що були визначені класифікатором, точність класифікації зростає до 86,15%.

Для оцінки некоректно класифікованих статей доцільно виконати зворотну оцінку: статті, для яких категорія, зазначена автором, не знаходиться серед найбільш ймовірних за оцінкою класифікатора швидше за все класифіковані неправильно. Для пошуку таких статей була використана наступна умова: категорія, зазначена автором не входить в топ-10 за оцінкою класифікатора, або ймовірність цієї категорії більш ніж на порядок відрізняється від найбільш ймовірної категорії. Такому критерію відповідає 4,82% статей. Ручна перевірка показала, що вони дійсно не відповідають вказаним категоріям.

При цьому статті, що не увійшли до першого або другого класу або мають нечітку тематику, яку можна віднести до багатьох категорій, або належать до помилок класифікатора. Підсумок який показує кількість неправильно класифікованих статей, що знаходяться в бібліотеці наведено на рисунку 4.2.

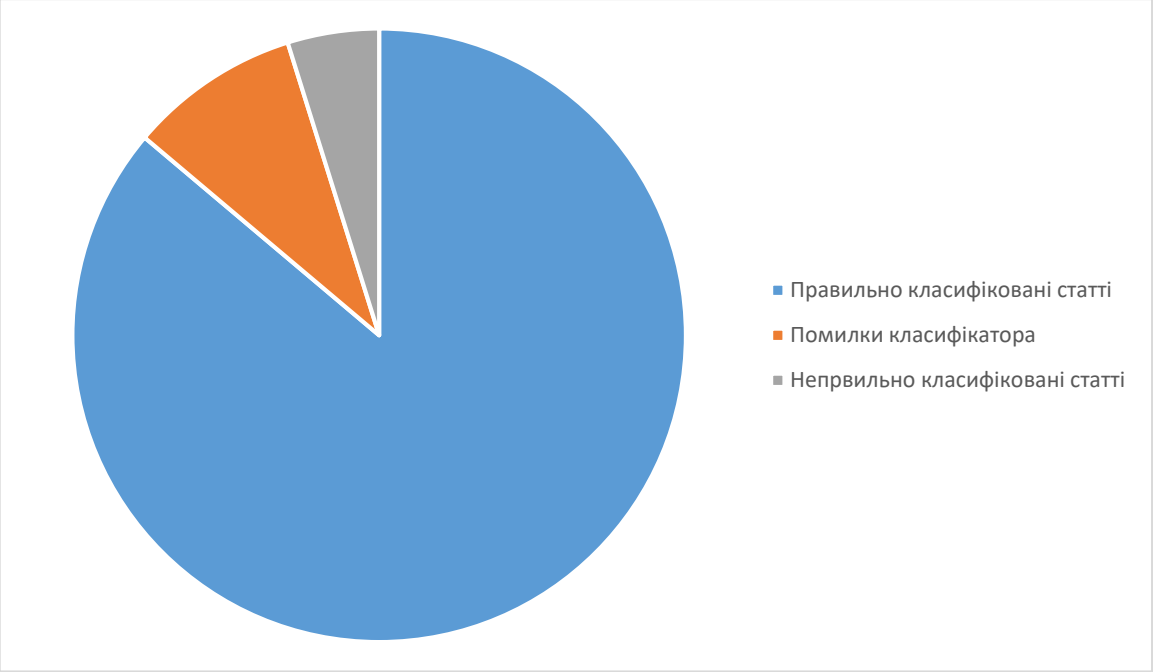

Рисунок 4.2 – Якість ручної класифікації

4.6 Висновки

В даній роботі наведено результати класифікації текстів за УДК. Найбільшу ефективність досягнуто за рахунок виконання ієрархічної класифікації з використанням нейронних мереж. Такий підхід дозволив досягти 59,74% загальної точності при класифікації текстів за двадцятьма найбільш популярними категоріями УДК.

Побудова класифікатора, який зможе виконувати класифікацію з достатньою для практичного застосування точністю потребуватиме створення більшої кількості моделей для покращення точності в суміжних категоріях та класифікації на глибших рівнях УДК. Також залишається відкритим питання збору навчальних даних для категорій третього рівня, адже в них тематика текстів стає дуже близькою, а статей бібліотеки Вернадського може виявитись недостатньо для тренування класифікатора до рівня, на якому він зможе визначати відмінності між настільки схожими статтями.

5 РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

Класифікацію текстів можна використати для категоризації статей, що будуть цікавими користувачу. На нашому веб-сайті користувач може вказати статті, що йому сподобались та, за бажанням, вказати категорії за якими він хоче отримувати статті. Таким чином він отримає можливість зручно отримувати цікаву інформацію, тому що йому не доведеться вручну шукати цікаві статті, а достатньо буде перейти на створений веб-сайт та вказати цікаві теми. Загальну ідею стартап проекту описано в таблиці 5.1.

Таблиця 5.1 - Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Класифікація текстів за категоріями	Класифікація статей за категоріями	Отримання статей тільки за цікавою тематикою
	Можливість задати власні категорії для класифікації	Можливість вказати власні категорії, на які будуть сортуватись статті
Оцінка подібності текстів на основі класифікації	Пошук подібних текстів до заданих	Можливість знайти статті за шуканою темою

Для оцінки актуальності продуту розглянемо альтернативні сервіси. Основними конкурентами даного продукту є:

- агрегація новий у пошуковій системі Google (Google Scholar);
- профільні сайти (наприклад opec.ru або habr.com).

Для розуміння конкурентоспроможності продукту опишемо відмінності від альтернативних сервісів. Порівняння функцій наведено в таблиці 5.2.

Таблиця 5.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	*Техніко-економічні характеристики ідеї*	*Концепції конкурентів*			*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Google Scholar*	*Профільні сайти*			
1	Вартість обслуговування серверів (грн./тис. користувачів)	80	40	30	✓		

Продовження таблиці 5.2

№ п/п	*Техніко-економічні характеристики ідеї*	*Концепції конкурентів*			*W (слабка сторона*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Google Scholar*	*Профільні сайти*			
2	Кількість новин, прочитаних користувачем за одну сесію	10	3	5		✓	
3	Зручність інтерфейсу (по п’ятибальній шкалі)	4	3	4		✓	
4	Зручність класифікації статей (по п’ятибальній шкалі)	5	2	1			✓
5	Схема монетизації	Реклама та промо-матеріали	Реклама	Реклама та промо-матеріали		✓	

Як бачимо, основними відмінностями між пропонованим сервісом та існуючими продуктами є те, що новий сервіс пропонує значно більшу зручність пошуку цікавих статей ціною збільшення собівартості обслуговування.

5.1 Технологічний аудит ідеї проекту

Проект складається з двох основних складових – рушій для класифікації та пошуку статей та веб-сайт, який надає користувачам зручний інтерфейс для перегляду та пошуку статей. Потенційні технології, що можуть використовуватись для реалізації проекту наведено в таблиці 5.3.

Таблиця 5.3 - Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1	Класифікація текстів	Рекурентні нейронні мережі в короткотривалою пам’яттю	Для розробки доступні бібліотеки Keras та TensorFlow	TensorFlow може вільно використовуватись в комерційних цілях
2	Створення веб-сайту для користувачів	Стандартна бібліотеки для створення веб-сайтів: Django	Технології готові для використання	Django може вільно використовуватись в комерційних цілях

Для класифікації текстів використовуються нейронні мережі, для роботи з якими використовуватись бібліотека deeplearning4j, розроблена компанією Google. На даний момент вона є провідним засобом для роботи з нейронними мережами. Крім того, вона є повністю безкоштовною, в тому числі для використання у комерційних цілях.

Для створення веб-сайту обрано бібліотеку Ignite. ЇЇ перевагами є можливість зручно та швидко створити веб-інтерфейс для додатку. Ця бібліотека створена на мові Java, так само як і deeplearning4j, що дозволяє їм запускатись в одному середовищі без додаткових витрат на передачу даних. Крім того, як і deeplearning4j, ця бібліотека є вільно розповсюджуваною, та може безкоштовно використовуватись для комерційних цілей.

Всі необхідні технології вже створені та є у вільному доступі ліцезновані для комерційного використання, що спрощує розроблення сервісу та зменшує собівартість його створення.

5.2 Аналіз ринкових можливостей запуску стартап-проекту

Для оцінки привабливості проекту оцінимо поточний стан ринку. Його характеристики наведені в таблиці 5.4.

Таблиця 5.4 - Попередня характеристика потенційного ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1	Кількість головних гравців, од	Немає
2	Загальний обсяг продаж, грн	4 344 млн.
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	150%

Бачимо, що ринок є привабливим, адже він має великий обсяг та зростає. Крім того, на ньому відсутні будь-які обмеження щодо стандартизації або

сертифікації, що робить вхід на ринок дуже легким. Також на цьому ринку всі користувачі поділені між невеликими компаніями, що дає змогу просто набрати початкову аудиторію та спрощує маркетинг.

Розглянемо потенційну аудиторію нашого проекту. ЇЇ основні риси наведені в таблиці 5.5.

Таблиця 5.5 - Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1	Отримання точної та актуальної інформації	Користувачі, які глибоко цікавляться певною тематикою: науковці, співробітники дослідних підрозділів, аналітики та консультанти	Бажання отримувати точну та актуальну інформацію у вибраній галузі	Оперативне отримання актуальних статей
2	Отримання цікавих статей у зручному форматі	Пересічні користувачі з доступом до мережі інтернет	Бажання отримати задоволення від прочитання цікавих статей	Зручність та цікавість отримання інформації

Потенційні фактори, що можуть загрожувати проекту або ускладнити його реалізацію вказані в таблиці 5.6.

Таблиця 5.6 - Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	*2*	*3*	*4*
1	Низька точність класифікації	Користувачі отримують нецікаві та нерелевантні новини, внаслідок чого припиняють користування сервісом	Зменшення кількості категорій класифікації, вимкнення можливості створення власних категорій
2	Поява аналогічного сервісу	Користувачі починають користуватись альтернативним сервісом	Вкладання додаткових коштів у маркетинг, розробка додаткових унікальних послуг

Продовження таблиці 5.6

1	*2*	*3*	*4*
3	Зростання вартості CUDA обчислень	Через зростання вартості обчислень падає рентабельність сервісу	Спрощення нейронних мереж для зменшення необхідної кількості обчислень, призводить до проблеми з п.1
4	Прийняття законів, що ускладнюють або забороняють агрегацію новин	Обмежується можливість отримання актуальних новин для надання їх користувачам	Укладання договорів з ЗМІ для отримання можливості агрегування їх новин

Основними загрозами є низька точність класифікації або зростання собівартості обчислень, що призведе до вимушеного зменшення кількості категорій або зростання вартості обслуговування системи. Поява аналогічного сервісу або консолідація кількох наявних може призвести до втрати користувачів. Крім того, на поточний момент робота агрегаторів статей ніяк не обмежена законодавчо. Але такі обмеження можуть бути створені: наприклад, в Іспанії є «податок на цитування», а введення аналогічного закону розглядає Єврокомісія з питань єдиного цифрового ринку.

Також розглянемо фактори, що можуть спростити реалізацію проекту (таблиця 5.7).

Таблиця 5.7 - Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Зростання попиту на агрегацію статей в одному місці	Збільшення потенційної аудиторій користувачів	Посилення маркетингу для перетворення потенційних користувачів на реальних
2	Здешевлення CUDA обчислень/поява оптимальніших алгоритмів	Зниження собівартості обчислень	Можливість збільшити точність класифікації та/або кількість класів

Основними можливостями є здешевлення обчислень та поява оптимальніших алгоритмів, що знизить собівартість роботи сервісу та дозволить надавати розширені послуги користувачам, наприклад, більшу кількість категорій або зростання точності класифікації.

Для оцінки конкурентоспроможності у таблиці 5.8 оцінимо загальні риси конкуренції на ринку.

Таблиця 5.8 - Ступеневий аналіз конкуренції на ринку

№ п/п	*Особливості конкурентного середовища*	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1	Вказати тип конкуренції: конкурентний	На ринку присутні багато профільних сайтів, що розміщують у себе статті, проте жоден з них не займає відчутну частку ринку	Немає необхідності боротися за користувачів з монополістами, тому цей фактор не є значущим
2	За рівнем конкурентної боротьби: національний	Користувачі переважно читають новини рідною мовою	Для ринку кожної країни необхідно буде адаптуватись, тому оптимально буде розпочинати роботу тільки в одній країні, а вже потім розповсюджуватись на інші ринки.
3	За галузевою ознакою: внутрішньогалузева	Основною альтернативою для сайту є інший сайт, отримати користувача, який надає перевагу іншому типу отримання інформації (телебачення, радіо) складно	Немає сенсу звертати увагу на користувачів, що не користуються інтернетом для отримання інформації, тому що вартість отримання такого користувача буде занадто високою
4	Конкуренція за видами товарів: товарно-видова	Аналогічно до попереднього пункту, отримати користувача, який не користується інтернетом дуже важко	Необхідно правильно презентувати свій сервіс, щоб користувачі знали про всі його переваги
5	За характером конкурентних переваг: нецінова	В першу чергу користувачів цікавить якість та повнота отриманої інформації	Необхідно створити зручний сервіс та правильно подати його користувачам
6	За інтенсивністю: марочна	Користувачі слідкують за джерелом інформації та надають перевагу відомим авторам та ЗМІ з гарною репутацією	Необхідно розвивати власний бренд та слідкувати за його репутацією

На ринку наявна велика кількість невеликих компаній, що спрощує вихід на ринок, адже немає необхідності боротися з монополістом, що цей ринок утримує. Оскільки користувачі читають статті переважно рідною мовою, то оптимальним буде оптимізація класифікації під певну мову та вихід на

конкретний ринок, а вже після досягнення успіху додавати підтримку інших мов. Оскільки користувачі переважно не міняють джерело отримання інформації, то маркетингові заходи доцільно проводити в інтернеті, бо, наприклад, конверсія з реклами на телебаченні буде надзвичайно низькою. За тієї ж причини необхідно чітко позиціонувати сервіс, щоб пояснити користувачам чому необхідно використовувати саме його, адже основним джерелом нових користувачів будуть конкурентні інтернет сайти. Оскільки рівень цін на ринку в середньому однаковий (більшість сервісів монетизується за рахунок реклами), а користувачі звертають увагу на джерело інформації та надають перевагу авторитетним джерелам, є необхідність у створенні гарної репутації для свого бренду.

Проведемо детальний аналіз умов конкуренції у галузі, звернувши увагу на учасників ринку (за моделлю п’яти сил М. Портера). Результати аналізу наведені в таблиці 5.9.

Таблиця 5.9 -Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
Складові аналізу	*Google Scholar, профільні інформаційні сайти*	*Необхідність створення моделей для точної класифікації новин; створення списків авторитетних ЗМІ для отримання інформації*	*При поточному стані ринку постачальники (інтернет ЗМІ) не мають ніякого впливу на агрегатори статей*	*Користувачам важлива точність та релевантність отримуваної інформації*	*Отримання новин з інших джерел інформації (телебачення, газети тощо)*
Висновки:	Всі вони не надають достатньої точності класифікації новин, тому треба зосередитись саме на цій функції	Поріг входу на ринок не надто високий, основною перевагою є створення точних моделей для надання користувачам актуальної інформації	На даний фактор можна не зважати	Необхідно забезпечити високу точність класифікації	Кількість користувачів, що змінюють джерело отримання інформації занадто мала, щоб на неї зважати

На основі проведеного аналізу можемо зробити висновок, що на ринку наявно багато розрізнених сервісів, тому є необхідність у чіткому позиціонуванні та акценті на перевагах власного сервісу. Оскільки бар'єри для входу на ринок практично відсутні, то є необхідність у отриманні лояльної аудиторії, яка отримає всі необхідні їй функції та не буде шукати інші сервіси, які потенційно можуть з'явитись на ринку.

На основі проведеного аналізу ринку та основних його учасників проведемо обґрунтування факторів конкурентоспроможності. Результати обґрунтування наведені в таблиці 5.10.

Таблиця 5.10 - Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування*
1	Точність класифікації	Висока точність класифікації новин надає можливість надання користувачу найбільш релевантної інформації, внаслідок чого збільшується лояльність користувачів
2	Можливість задавати власні категорії для класифікації	Користувачі отримують можливість читати новини не тільки за наперед заданими категоріями, але й задавати власні критерії для їх сортування, що збільшує релевантність отримуваної інформації

Основними функціями, що дають можливість отримати нових користувачів на ринку та їх утримання у разі появи конкурентів є надання унікальних функцій. Такими функціями є можливість класифікації статей, що дозволяє легко знаходити інформацію за власними інтересами. Основним фактором для утримання користувачів є можливість вказати статті, що їм сподобались та отримувати подібні статті.

За визначеними факторами конкурентоспроможності, а також на основі факторів загроз та можливостей (таблиці 5.6 та 5.7) проведемо SWOT-аналіз проекту в таблиці 5.11.

Таблиця 5.11 - SWOT-аналіз стартап-проекту

Сильні сторони: висока точність класифікації, можливість створення власних критеріїв для пошуку новин	Слабкі сторони: низька початкова аудиторія, необхідність великої кількості початкових даних для отримання достатньої точності класифікації
Можливості: покращення сервісу в разі появи оптимізованого програмного та апаратного забезпечення для обчислень	Загрози: поява на ринку великого гравця (наприклад Google), збільшення вартості обчислень, ускладнення отримання новин

Сильними сторонами проекту є унікальність функцій, що пропонуються користувачам. Слабкими сторонами є високі вимоги для створення точних класифікаторів: великий обсяг навчальних даних та висока складність обчислень.

Для оцінки стратегій виходу на ринок розробимо альтернативи ринкового впровадження. Альтеративні алгоритми поведінки наведено в таблиці 5.12.

Таблиця 5.12 - Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1	Створення мінімального продукту та швидкий вихід на ринок	Висока	Від шести до восьми місяців
2	Створення потужного математичного апарату та вихід на ринок тільки після досягнення високої точності класифікації	Середня	Два роки

Доступні дві стратегії – створення мінімального продукту для виходу на ринок для оцінки реакції користувачів, та створення продукту, що буде містити всі бажані функції, але з відкладеним виходом на ринок. Обраною альтернативою є перша, адже є важливим отримання зворотного зв’язку від користувачів перед вкладанням ресурсів у функції, які можуть в подальшому не знадобитись. Також, збільшення термінів розробки збільшує ймовірність появи конкурентів зі схожими функціями.

5.3 Розроблення ринкової стратегії проекту

Для створення ринкової стратегії проаналізуємо цільові групи потенційних користувачів. Характеристики користувачів наведені в талблиці 5.13.

Таблиця 5.13 - Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1	Профільні спеціалісти	Низька	Високий	Низька	Низька
2	Пересічні люди з доступом до мережі інтернет	Висока	Середній	Висока	Висока
Які цільові групи обрано: в першу чергу треба орієнтуватись на пересічних користувачів, оскільки вони готові до сприйняття продукту, а вхід не потребує значних ресурсів.					

Отже, в першу чергу проект орієнтований на масову аудиторію, адже вона легко сприймає нові сервіси.

На основі отриманої інформації та альтернативи ринкового впровадження стартап-проекту розробимо базову стратегію розвитку. Розроблена стратегія підсумована в таблиці 5.14.

Таблиця 5.14 - Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку**
1	Вихід на масовий ринок з сервісом зручного пошуку цікавих статей	Масовий маркетинг	Можливість зручно отримувати тільки цікаві статті; наявність статей різного профілю в одному місці	Стратегія диференціації

Базовою стратегією розвитку є вихід на масовий ринок з потужним масовим маркетингом. Основним акцентом в маркетинговій стратегії є виділення ключових рис сервісу, а саме можливість зручно отримувати тематичні статті.

На основі базової стратегії розвитку розробимо стратегію конкурентної поведінки. Розроблена стратегія підсумована в таблиці 5.15.

Таблиця 5.15 - Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки**
1	Частково. На ринку вже є багато сервісів для отримання новин, але жоден з них не дає можливості зручно класифікувати інформацію	Компанія буде забирати споживачів у вже існуючих ЗМІ.	Компанія не буде копіювати конкурентів	Зайняття конкурентної ніші

Даний стартап проект не є першопроходьцем на ринку, а тому йому доведеться забирати користувачів у вже існуючих сервісів. Оскільки ідея динамічної класифікації статей з’являється на ринку вперше, необхідно чітко визначити свої переваги та зайняти власну нішу.

Визначимо стратегію позиціонування з основою на акценті унікальних рис розроблюваного сервісу. Розроблена стратегія підсумована в таблиці 5.16.

Таблиця 5.16 - Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
1	Релевантність отримуваних статей	Стратегія диференціації	Точність класифікації новин на основі переваг користувачів	Актуальність, точність, релевантність
2	Зручність пошуку актуальної інформації	Стратегія диференціації	Можливість вводити власні критерії для класифікації	Зручність, персоналізація

Отже, основою ринкової стратегії буде отримання користувачів, що користуються проектами конкурентів. Обраною стратегією буде стратегія диференціації з акцентом на унікальності пропонованих функцій.

5.4 Розроблення маркетингової програми стартап-проекту

На основі проаналізованих ознак конкурентоспроможності товару розробимо ключові переваги потенційного товару. Основні переваги наведені в таблиці 5.17.

Таблиця 5.17 - Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами*
1	Отримання цікавих та релевантних новин	Динамічна система класифікації, що дає можливість швидко отримувати релевантні новини	Схожа система є лише у Twitter, але наший сервіс пропонує повноцінні статті на відміну від Twitter, який оперує короткими повідомленнями
2	Зручність пошуку цікавої інформації	Можливість вказати статті що сподобались та отримати рекомендації на основі них	Аналогів на ринку текстових статей поки що немає

Так, основними перевагами перед конкурентами є отримання релевантних новин не за визначеними категоріями, а за персональними вподобаннями користувача. Також сервіс надає можливість пошуку подібних статей до заданих, що спрощує дослідження специфічної тематики.

Опишемо особливості розроблюваного продукту в таблиці 5.18.

Таблиця 5.18 - Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*
I. Товар за задумом	Отримання цікавих та актуальних статей у зручному для користувача форматі
II. Товар у реальному виконанні	Властивості/характеристики
	1. Доступний у вигляді веб-сайту на комп'ютерах та мобільних пристроях 2. Можливість створювати власні категорії для класифікації статей 3. Можливість самому написати статтю
	Марка: назва організації-розробника + назва товару
III. Товар із підкріпленням	Можливість зберігати статті, щоб повернутися до читання пізніше; можливість ділитися посиланнями на цікаві статті у соціальних мережах; можливість підписуватись на оновлення за певною тематикою
За рахунок чого потенційний товар буде захищено від копіювання: унікальність алгоритмів класифікації, що становить основу сервісу.	

Отже основною ідеєю сервісу є можливість отримання цікавих та актуальних статей у зручному для користувача форматі. Крім того, сервіс надає

можливості створювати власні категорії для класифікації статей, а також можливість написання статей. Повернення користувачів відбувається за рахунок персоналізації статей, можливість їх збереження, можливість поділитися посиланням на цікаву статтю в соціальних мережах, а також можливість підписуватись на оновлення за певною темою. Захист від копіювання буде відбуватись за рахунок унікальності алгоритмів класифікації, що є основною перевагою сервісу.

На основі рівня цін на товари-замінники, товари-аналоги, а також зважаючи на рівень доходів визначимо межі на встановлення цін в таблиці 5.19.

Таблиця 5.19 - Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники*	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
1	В середньому ціна на онлайн рекламу становить від 25 до 100 грн. на тисячу показів. Ведення корпоративних блогів 10-50 тис. грн.	Вартість випуску рекламного ролику на телебаченні – 30-100 тис. грн. при аудиторії в 2 млн. осіб	Невеликі IT компанії мають виручку близько 1 млн. грн., для великих середній дохід з користувача становить близько 500-600 грн. при конверсії в 2.3%	Нижня межа – 25 гривень на тисячу показів, верхня – до 2000 гривень

Основними джерелами монетизації буде розміщення рекламних матеріалів. Точна ціна буде залежати від популярності сайту, основними межами є від 25 до 2000 гривень за тисячу рекламних показів.

Проаналізуємо систему збуту та визначимо глибину каналу збуту та потенційних посередників. Основні канали збуту наведено в таблиці 5.20.

Таблиця 5.20 - Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1	При продажу реклами клієнти переважно самостійно шукають площадки. Але можливим є	Для послуг інтернет-реклами немає потреби в проміжній ланці постачальника, зазвичай рекламні	Мінімальна – контактувати напряму з замовником реклами	Отримати популярність та розмістити інформацію про можливість

	приєднання до медіа концернів.	матеріали передаються напряму.		реклами на власному сайті

Отже оптимальним є прямий контакт із замовником реклами, адже надання електронних послуг не містить складних логістичних чи інформаційних проблем, за яких знадобилася б допомога посередника. Тем не менш, на початковому етапі розвитку при відсутності власних замовників доцільним є приєднання до великого постачальника реклами.

На основі сформованих переваг продукту визначимо завдання маркетингових комунікацій. Розроблену маркетингову стратегію наведено в таблиці 5.21.

Таблиця 5.21 - Концепція маркетингових комунікацій

№ п/п	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуються цільові клієнти*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*
1	В інтернеті є велика кількість сервісів, тому користувач в першу чергу звертає увагу на відмінність пропонованого сервісу від інших	Переважно інтернет	Можливість централізовано читати цікаві статті та легко знаходити необхідні матеріали	Донести інформацію про можливість зручно отримувати статті за інтересами	Розповісти про можливість отримати будь-яку інформацію за своїм бажанням

Отже, основними задачами маркетингових комунікацій є акцентування на сильних сторонах проекту, а саме на можливості отриманні статей за власними інтересами та можливості легкого пошуку подібної інформації. Основним каналом реклами має бути інтернет, адже саме ним користується переважна частина потенційних користувачів.

5.5 Висновки

Основною ідеєю розроблюваного проекту створення веб-сайту на якому користувачі можуть виконувати пошук цікавих статей. Замість постійного

моніторингу виходу статей на профільних сайтах користувач може просто вказати свої вподобання та отримати релевантні статті.

Основними технологіями, що будуть використані для створення проекту є бібліотеки TensorFlow та Django, створені для мови програмування Python. TensorFlow це бібліотека для машинного навчання, що створена компанією Google. Django надає можливість швидко та просто створювати веб-сайти. Спільними перевагами цих бібліотек є спільне середовище, що спрощує передачу класифікованих даних на веб-сайт, а також безкоштовність та відсутність обмежень на комерційне використання.

На обраному ринку відсутні великі гравці та обмеження для входу (стандартизація, сертифікація тощо). Тим не менш він є досить великим (станом на 2018 рік він оцінюється в 4,5 млрд грн) та постійно зростає. Основним видом конкуренції на ринку є внутрішньогалузева, що означає необхідність отримання користувачів конкурентів. Оскільки на ринку переважає нецінова марочна конкуренція, то важливим завданням маркетингу є акцент на особливостях сервісу та його відмінностях від конкурентів а також створення позитивної репутації та її моніторинг.

Головними факторами конкурентоспроможності є зручність отримання цікавих та актуальних статей, а також можливість тематичного пошуку на основі заданого тексту. Слабкостями проекту є більша собівартість на обслуговування користувачів, що може зробити проект нерентабельним через зростання цін на обчислення, проте при виникненні такої ситуації можна зменшувати кількість категорій для класифікації до тих пір, поки вартість обчислень не повернеться до прийнятного рівня. Захищеність від потенційних конкурентів полягає у складності розробки ефективних алгоритмів класифікації, що становлять основу функцій сервісу.

Потенційною аудиторією проекту є молода аудиторія, що користується інтернетом. Цей сегмент аудиторії має високу готовність до сприймання нових продуктів, а складність входу в сегмент є невисокою. Після освоєння цього сегменту можна розглянути можливість просування продукту для профільних

спеціалістів. Ця аудиторія є менш готовою для сприйняття нового продукту, але й більш лояльною. Для подальшого розширення аудиторії можна розглянути вихід на іншомовні ринки, що потребує модифікації алгоритмів класифікації для інших мов.

Основною стратегією позиціонування є стратегія диференціації з акцентом на основні відмінності від конкурентів, а саме можливість отримання релевантних статей за власними інтересами. Каналом для просування проекту є інтернет, адже люди рідко міняють джерело отримання інформації, тому реклама, наприклад, на телебаченні, буде неефективною.

Монетизація проекту буде відбуватись за рахунок продажу реклами. Межами ціни на рекламу є 25-2000 гривень на тисячу рекламних показів. Найбільш оптимальним є прямий контакт з рекламодавцем, проте на початковому етапі роботи проекту можливе використання постачальника реклами (наприклад Google AdWords).

У підсумку даний проект є перспективним. Ринок є досить великим та постійно зростає, на ньому немає відчутних вхідних бар’єрів та монополістів. Хоча користувачі на ринку вже поділені між існуючими виданнями, проте за умови надання унікального (для залучення нових користувачів) та зручного (для їх утримання) сервісу може принести значну вигоду.

ВИСНОВКИ

В даній роботі було досліджено методи класифікації текстів, написаних українською мовою. У якості методів класифікації було використано наївний баєсів класифікатор та нейронні мережі.

Дані для навчання класифікаторів були завантажені з наукового архіву бібліотеки ім. Вернадського. Класифікація виконувалась по двадцяти найбільш популярним категоріям УДК першого рівня. Всього було використано 70170 статей, з яких 70% застосовані для навчання, а 30% для тестування класифікаторів. Незважаючи на малу кількість навчальних даних нейронна мережа показала вищу ефективність порівняно з наївним баєсовим класифікатором (з точністю в 47,66% проти 45,70% відповідно). Проте, найбільш ефективним є використання ієрархічного класифікатора, який класифікує тексти з точністю в 60%.

В результаті створений класифікатор може використовуватись для пошуку помилок в текстах, що були класифіковані вручну, а також як допоміжний засіб при класифікації нових надходжень.

В подальшому для практичного застосування класифікатора необхідно покращити точність класифікації суміжних категорій. Також необхідно зібрати дані для навчання класифікаторів на УДК третього рівня, тому що текстів бібліотеки ім. Вернадського виявилося надто мало.

Для роботи зі створеними класифікаторами було створено програмне забезпечення, яке підтримує як графічний, так і програмний інтерфейс, що дає можливість частково або повністю автоматизувати процес класифікації наукових статей. Також було виконано оцінку комерційної цінності створеного програмного забезпечення та розроблено стратегію створення стартап проекту.

ПЕРЕЛІК ПОСИЛАНЬ

1) Alvarez, S. A. (2002). An exact analytical relation among recall, precision, and classification accuracy in information retrieval. Boston College, Boston, Technical Report BCCS-02-01, 1-22.

2) Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In Mining text data (pp. 163-222). Springer, Boston, MA.

3) Korde, V., & Mahender, C. N. (2012). Text classification and classifiers: A survey. International Journal of Artificial Intelligence & Applications, 3(2), 85.

4) Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural networks, 61, 85-117.

5) Gaigole, P. C., Patil, L. H., & Chaudhari, P. M. (2013). Preprocessing techniques in text categorization. In National Conference on Innovative Paradigms in Engineering & Technology (NVIPET-2013), Proceedings published by International Journal of Computer Applications (IJCA).

6) Bergmanis, T., & Goldwater, S. (2019). Data Augmentation for Context-Sensitive Neural Lemmatization Using Inflection Tables and Raw Text. arXiv preprint arXiv:1904.01464.

7) Anandarajan, M., Hill, C., & Nolan, T. (2019). Text Preprocessing. In Practical Text Analytics (pp. 45-59). Springer, Cham.

8) Kulkarni, A., & Shivananda, A. (2019). Exploring and Processing Text Data. In Natural Language Processing Recipes (pp. 37-65). Apress, Berkeley, CA.

9) Scott, S., & Matwin, S. (1998). Text classification using WordNet hypernyms. Usage of WordNet in Natural Language Processing Systems.

10) Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

11) McCallum, A., & Nigam, K. (1998, July). A comparison of event models for naive bayes text classification. In AAAI-98 workshop on learning for text categorization (Vol. 752, No. 1, pp. 41-48).

12) Csáji, B. C. (2001). Approximation with artificial neural networks. Faculty of Sciences, Etvs Lornd University, Hungary, 24, 48.

13) McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.

14) Specht, D. F. (1990). Probabilistic neural networks. Neural networks, 3(1), 109-118.

15) Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. IEEE transactions on neural networks, 8(1), 98-113.

16) Amin, M. Z., & Nadeem, N. (2018). Convolutional Neural Network: Text Classification Model for Open Domain Question Answering System. arXiv preprint arXiv:1809.02479.

17) Ханько, Г. В. Методи класифікації текстових даних для виявлення пропаганди : магістерська дис. : 126 Інформаційні системи та технології / Ханько Ганна Вадимівна. – Київ, 2018. – 79 с.

18) Werbos, P. J. (1974). New tools for Prediction and Analysis in the Behavioral Sciences. Ph. D. dissertation, Harvard University.

19) Mishkin, D., Sergievskiy, N., & Matas, J. (2016). Systematic evaluation of CNN advances on the ImageNet. arXiv 1606: 02228 [cs]. arXiv preprint arXiv:1606.02228

20) Kumar, S. K. (2017). On weight initialization in deep neural networks. arXiv preprint arXiv:1704.08863.